EXPONENTIAL CHARACTERISTIC SPATIAL QUADRATURE FOR DISCRETE ORDINATES NEUTRAL PARTICLE TRANSPORT IN SLAB GEOMETRY

THESIS

Glenn E. Sjoden, Captain, USAF

AFIT/GNE/ENP/92M-10

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GNE/ENP/92M-10

EXPONENTIAL CHARACTERISTIC SPATIAL QUADRATURE FOR DISCRETE ORDINATES NEUTRAL PARTICLE TRANSPORT IN SLAB GEOMETRY

THESIS

Glenn E. Sjoden, Captain, USAF

AFIT/GNE/ENP/92M-10

AFIT/GNE/ENP/92M-10

EXPONENTIAL CHARACTERISTIC SPATIAL QUADRATURE FOR DISCRETE ORDINATES NEUTRAL PARTICLE TRANSPORT IN SLAB GEOMETRY

THESIS

Presented to the Faculty of the School of Engineering

of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the

Requirements for the Degree of

Master of Science in Nuclear Engineering

Glenn E. Sjoden, B.S., P.E.

Captain, USAF

March 1992

Preface

Despite recent advances in computer hardware, the innovation of new, efficient numerical algorithms still yields the maximum gain in analyzing problems once thought to be too large, too difficult, or too expensive to solve. I believe the exponential characteristic discrete ordinates scheme, developed here in slab geometry, demonstrates this theme extremely well. Expansion of this scheme to more complex geometries should produce even more remarkable results.

I would like to thank my advisor, LCDR K. Mathews, Ph.D. for suggesting this research topic and providing me with background materials and existing computer codes. His guidance and instinctive numerical methods expertise were essential in completing this project. Also, my heartfelt gratitude goes to my loving wife Patti, who faithfully supported me, and to God for granting me the wisdom and patience to finish this project.

Table of Contents

List of Figures

List of Tables

Abstract

A new discrete ordinates spatial quadrature scheme is presented for solving neutral particle transport problems. This new scheme, called the exponential characteristic method, is developed here in slab geometry with isotropic scattering. This method uses a characteristic integration of the Boltzmann transport equation with an exponential function as the assumed form of the source distribution, continuous across each spatial cell. The exponential source function is constructed to globally conserve zeroth and first spatial source moments and is non-negative. Characteristic integration ensures non-negative fluxes and flux moments. Numerical testing indicates that convergence of the exponential characteristic scheme is fourth order in the limit of vanishingly thin cells.

Highly accurate solutions to optically thick problems can result using this scheme with very coarse meshes. Comparing accuracy and computational cost with existing spatial quadrature schemes (diamond difference, linear discontinuous, linear characteristic, linear adaptive, etc.), the exponential characteristic scheme typically performed best. This scheme is expected to be expandable to two dimensions in a straight forward manner. Due to the high accuracies achievable using coarse meshes, this scheme may allow researchers to obtain solutions to transport problems once thought too large or too difficult to be adequately solved on conventional computer systems.

I. Introduction

Neutral particle transport methods are often used to solve complex problems in nuclear science and engineering. Several Air Force programs routinely require detail. d numerical models of neutron and photon transport in nuclear systems, including space nuclear power reactors and nuclear effects experiments. The innovation of new, more efficient numerical schemes for solving radiation transport problems can be beneficial, often providing very accurate results while consuming fewer resources than required by conventional numerical transport methods. Accurate modeling of neutrons and gamma rays in a given system can be achieved using a discrete ordinates solution to the Boltzmann transport equation, which is given below in the general integro-differential form for neutrons (Lewis and Miller, 1984:34-39):

$$\frac{1}{v}\frac{\partial\psi(\vec{r},\hat{\Omega},E,t)}{\partial t}+\hat{\Omega}\cdot\vec{\nabla}\psi(\vec{r},\hat{\Omega},E,t)+\sigma(\vec{r},E)\psi(\vec{r},\hat{\Omega},E,t)= s_{ext}(\vec{r},\hat{\Omega},E,t)+\int dE'\int d\hat{\Omega}'\sigma_s(\vec{r},E'\rightarrow E,\hat{\Omega}'\cdot\hat{\Omega},)\psi(\vec{r},\hat{\Omega}',E',t) +\chi(E)\int dE'\nu\sigma_f(\vec{r},E')\phi(\vec{r},E',t) \quad (1)$$

where

v = neutron speed

ψ = angular neutron flux

$\hat{\Omega}$ = unit vector in direction of particle motion

$\vec{r}$ = coordinate location in space

E = neutron energy

t = time

σ = total macroscopic cross section

s_{ext} = external neutron source

σ_s = scattering cross section causing neutrons to arrive in E and $\hat{\Omega}$

χ = fission neutron energy distribution function

$\nu\sigma_f$ = fission neutrons produced

ϕ = scalar neutron flux

Functional dependence of each variable is defined in the Boltzmann transport equation above. The right hand side of the transport equation is often referred to as the source function $S(\vec{r},\hat{\Omega},E,t)$, which incorporates the external, scattering, and fission source terms implicitly. A discrete ordinates solution to equation (1) consists of evaluating this equation in a number of distinct angular directions over a defined spatial mesh over a range of defined energy groups. Suitable angular and spatial quadratures are used to approximate integral particle fluxes, currents, and moments (Lewis and Miller, 1984:116-119).

This thesis introduces a new discrete ordinates spatial quadrature method to solve the Boltzmann transport equation in slab geometry. This approach, the "exponential characteristic" (EC) method, assumes an exponential form for the position dependence of the source function, defined over each cell in a way that preserves the zeroth and first spatial moments of the source over the cell. Cell edge fluxes and cell flux moments are obtained by using this exponential source representation in a characteristic integration of the Boltzmann transport equation.

In addition to having the qualities of positivity, continuity over a cell, and conservation of zeroth and first spatial source moments, numerical testing has demonstrated that this new exponential characteristic scheme has other desirable properties:

- Fourth order truncation error as $\sigma\Delta x \rightarrow 0$
- Proper solutions to diffusive problems, as tested
- Highly accurate over very coarse meshes

Ultimately, an efficient method that proves to be accurate in one dimension offers the most utility if easily adaptable to multidimensional coordinate systems so that more difficult problems might be solved. Based on applications of the exponential characteristic method in one dimension with isotropic scattering, presented here, this method should be generalizable to multi-dimensional geometries.

In Section II, necessary background information is discussed. The research problem is formally stated in Section III, with all applicable assumptions and limitations. Theoretical development of the exponential characteristic method is presented in Section IV, including methods for converting equations into stable computable forms. This is followed by computer program development and code validation in Section V. In Section VI, test results from exponential characteristic solutions to three test cases are evaluated and compared to other discrete ordinates schemes. Finally, conclusions and recommendations are given in Section VII, followed by several appendices.

II. Background

The transport equation can be solved only numerically for complex real world applications, e.g., reactor core and shielding problems; therefore, it is essential that any computer algorithm applied be efficient and accurate to a degree that affords a solution within a reasonable amount of time and cost. There are, in principle, two approaches to spatial quadrature discrete ordinates schemes used in solving the Boltzmann transport equation:

1) Characteristic schemes - based on the known streaming properties of the transport equation (e.g., an assumed form of the source function)

2) Curve fitting schemes - based on an assumed form of the flux distribution function $\psi(\vec{r})$

Regardless of which approach is used, five desirable properties of spatial quadrature schemes to be considered in solving the Boltzmann transport equation are:

1) accuracy - where there is a small truncation error

2) simplicity - where a small number of numerical operations with unknowns from a single mesh cell are involved

3) positivity - where positive flux values result from positive sources and positive entering boundary fluxes

4) particle conservation - where particles are neither created nor destroyed arbitrarily, with moment balances (e.g. zeroth moment, first moment, etc.) satisfied up to a given order

5) generalizability - where the method can be applied to other geometries: Cartesian, cylindrical, or spherical (Lathrop, 1969:475-477)

An additional quality for any scheme, introduced by Larsen, requires that for problems that are diffusive (absorptions and gradients of the angular flux are small), the solutions resulting from any given transport method must agree with solutions arrived at using the neutron diffusion equation. This criteria is defined as the diffusion limit. For some schemes, satisfying the diffusion limit is a problem, especially when the spatial mesh is optically thick (Larsen, 1983:90-98). It is difficult for a numerical scheme to meet all of the above criteria. For most schemes, maintaining positive fluxes throughout a problem is difficult unless optically thin cells are specified, which can result in very high computational costs (particularly in two or three space dimensions).

A. Diamond Difference Method

Positivity is sacrificed for simplicity in the implementation of the popular diamond difference method. Consider a one dimensional slab in the x direction made up of i cells, where any ith cell has length Δx_i (between x_{i-1} and x_i) with homogeneous material properties. Diamond difference is a curve fitting scheme where one assumes $\psi(x)_i$ is a linear function across a cell (Lathrop, 1969:482-484). In the case of steady-state, mono-energetic, non-multiplying, isotropic scattering and sources, Cartesian (slab) geometry, the Boltzmann transport equation simplifies to (suppressing the subscript i, and noting implicitly that each angular flux $\psi_m(x)$ is along a specific direction cosine μ_m):

$$\mu_m \frac{d\psi_m(x)}{dx} + \sigma_m \psi(x) = s(x) \tag{2}$$

where the source function is

$$s(x) = c\sigma\phi(x) + s_{ext}(x). \tag{3}$$

The scattering ratio in equation (3) is defined as

$$c = \frac{\sigma_s}{\sigma} \tag{4}$$

The scalar flux $\phi(x)$ for any spatial quadrature method, defined as the angular flux integrated over all angles, is given in slab geometry over all direction cosines μ between -1 and 1 (corresponding to all angles between π and 0 radians, respectively) by

$$\phi(x) = \int_{-1}^{+1} \psi(x,\mu)\frac{d\mu}{2} \tag{5}$$

Using the discrete ordinates method, the scalar flux is approximated by

$$\phi(x) = \frac{1}{2}\sum_{m=1}^{M} w_m \psi_m(x) \tag{6}$$

where M angular fluxes are evaluated at M discrete ordinates μ_m. Each direction cosine is from a quadrature set (w_m, μ_m). The eight direction cosines from an eight-point single range (S_8) Gauss-Legendre quadrature set, used for all computations in this thesis, are provided in Appendix A.

Equation (2) can be integrated over a single spatial (mesh) cell, with this cell defined locally between 0 and Δx. When $\mu > 0$, then particles flow from left to right, and $\psi(0) = \psi_L$ at $x = 0$ and $\psi(\Delta x) = \psi_R$ at $x = \Delta x$ (Note that L=Left, R=Right, and A=Average). This results in the zeroth moment balance equation:

$$\mu(\psi_R - \psi_L) + \sigma \Delta x \psi_A = S_A \Delta x \tag{7}$$

where the cell average source function and flux are

$$S_A = \frac{1}{\Delta x}\int_0^{\Delta x} s(x)dx \tag{8}$$

and

$$\psi_A = \frac{1}{\Delta x}\int_0^{\Delta x} \psi(x)dx \tag{9}$$

A single spatial cell is shown in Figure 1.

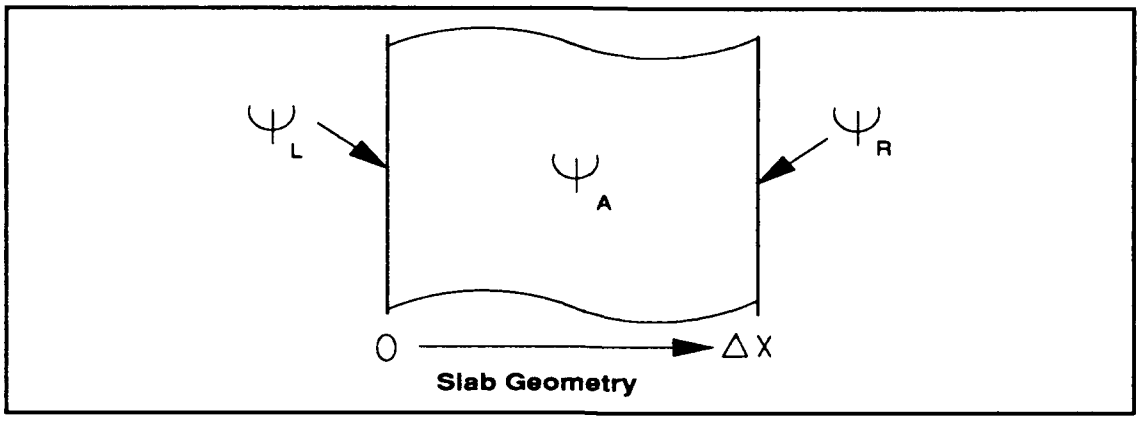

Figure 1. A Single Spatial Cell

The diamond difference method approximates equation (9), the cell average angular flux ψ_A (called the zeroth moment of the angular flux) by the arithmetic average of the left and right boundary fluxes,

$$\psi_A \approx \frac{1}{2}(\psi_L + \psi_R) \tag{10}$$

This is the fundamental assumption of the diamond difference method. Since ψ_L and S_A are either given from an initial guess or are available from a previous iteration, the basic diamond difference equation is obtained by combining equations (7) and (10):

$$\psi_R = \left(\frac{2-\varepsilon}{2+\varepsilon}\right)\psi_L + \left(\frac{2}{2+\varepsilon}\right)\frac{S_A \Delta x}{\mu} \tag{11}$$

where the angular optical thickness, in mean free paths, is defined as

$$\varepsilon = \left(\frac{\sigma \Delta x}{\mu} \right) \tag{12}$$

Ideally, the diamond difference method yields reasonable results and second order convergence in the limit of optically thin cells, where $\sigma \Delta x \rightarrow 0$ (Lewis and Miller, 1984:132). However, even if ψ_L and S_A are both positive, a negative angular flux is possible on the right cell boundary if the optical thickness is greater than two. Negative fluxes can occur even when the cell thickness Δx is small, because the cross section σ can be large, and the direction cosine μ can be small. In reality, negative fluxes have no physical meaning. In addition to the doubt negative fluxes cast on the usefulness of results, they can propagate unfavorably through a problem.

This can be treated by using a negative flux fixup, where if $\psi_R < 0$, then it is set to zero. The zeroth moment balance, equation (7), is then solved for ψ_A. While practical, these fixups can cause undamped oscillations among successive iterations, interfering with or even preventing convergence (Alcouffe, et al., 1979:111-115). In addition, negative flux fixups increase the truncation error of results, can cause failures in diffusion acceleration procedures, and can ultimately lead to incorrect solutions (Lathrop, 1969:475).

B. Other Spatial Quadrature Schemes

Following a wide use of the diamond difference method to solve transport problems (in spite of its inherent limitations), many researchers sought to develop alternative spatial quadrature schemes. In general, other methods that try to overcome the difficulties of diamond difference are less straight forward. Nevertheless, while some of these schemes are more intricate to encode, many achieve more accurate results than

diamond difference over a coarser spatial mesh. Therefore, although some alternative methods require increased computational effort per cell compared to diamond difference, the use of a coarser mesh usually lowers overall computation times and memory storage requirements (This is particularly important in two and three dimensional geometries). Some alternatives to diamond difference are listed in Table 1.

Table 1
Alternative Spatial Quadrature Schemes in Discrete Ordinates

1. (SC) Step Characteristic (Lathrop, 1969)
2. (LD) Linear Discontinuous (Hill, 1975)
3. (EM) Exponential Method (Barbucci, et al., 1977)
4. (LC) Linear Characteristic[+] (Alcouffe, et al., 1979)
5. (LN) Linear Nodal[+] (Walters, et al., 1981)
6. (LNA) Augmented Linear Nodal (Walters, 1986)
7. (SGF) Spectral Green's Function (DeBarros, et al., 1990)
8. (SA) Step Adaptive (Mathews, 1990)
9. (LA) Linear Adaptive (Mathews, 1990)
10. (MB) Multiple Balance (Morel, et al., 1990)
[+]Linear Nodal is identical to Linear Characteristic in slab geometry

Although most are improvements to diamond difference, each method in Table 1 has distinct advantages and disadvantages. Detailed descriptions of each scheme can be found in the literature. Some of the schemes listed in Table 1 are briefly mentioned here; others are left for discussion in later sections and will aid in the development of the exponential characteristic method.

The linear discontinuous (LD) method was developed in an attempt to diminish the number of negative flux fixups required by the diamond difference method. The fundamental assumption used in the LD method is that ψ is piecewise linear in x, but is discontinuous on each left (entering) cell boundary for $\mu > 0$, and discontinuous on each right (entering) cell boundary if $\mu < 0$. Similarly, the source is also taken to be piecewise

linear. These piecewise representations for the angular flux and source terms are introduced into zeroth and first spatial moments of the transport equation. This generates two equations solvable for two unknowns, the unknowns being the two discontinuous values of angular fluxes at a cell edge. The LD method is considered to be a near positive method, producing fewer negative fluxes than the diamond difference scheme for a given problem. Although negative fluxes can result, the linear discontinuous scheme does not invoke a fixup and achieves second order convergence in the limit of thin cells. The linear discontinuous scheme is algebraically equivalent to the linear nodal scheme if a Pade' (1,2) approximation is used for the exponential function in linear nodal. Because the LD scheme continues to be widely implemented in transport codes, it is used to aid in evaluating the performance of the exponential characteristic method (Hill, 1975), (Alcouffe, et al., 1979:117-118), and (Walters, 1981:115).

Of particular note is the exponential method (EM), developed by Barbucci and DiPasquantonio in 1977 (not to be confused with the exponential characteristic scheme developed in this effort). Recall that the diamond difference method, referring to equation (10), assumes the cell average angular flux is the arithmetic mean of the entering and exiting fluxes. Similarly, the exponential method, also a curve-fitting scheme, assumes the cell average flux is the geometric mean of entering and exiting fluxes, which results in an exponential functional form for all angular fluxes. This method, while positive, has fundamental difficulties in source regions and near vacuum boundaries. In addition, exact solutions are always overestimated, even in the limit of thin cells (Barbucci, et al., 1977). Because of these difficulties, no comparisons with the exponential method are made.

In slab geometry, there are no real differences between the linear characteristic method (LC) (discussed in a later section) and the linear nodal (LN) method, except in the implementation of a fixup (the reader is referred to the literature for treatment in other geometries). The LN method actuates a fixup if the magnitude of the first moment of the scalar flux exceeds that of the scalar flux zeroth moment, while the LC method implements a fixup based on the magnitudes of the source moments. In the limit of thin cells, LC and LN are equivalent in slab geometry. The augmented linear nodal scheme (LNA) implements the diamond difference relation for specific terms in two and three-dimensional moment balance equations to increase computational efficiency (with a small penalty in accuracy). In any case, the LNA method is not always positive, and a fixup is still required (Walters, et al., 1981), (Alcouffe, et al., 1979), and (Walters, 1986). Because LN is equivalent to LC in slab geometry, no other reference to it is made. Furthermore, as the LNA scheme is neither absolutely positive nor as accurate as the original LN scheme, it is not discussed further.

The spectral Green's function method (SGF) has no spatial truncation error in slab geometry. In addition to using the standard transport balance equations, it implements a non-standard Green's function for ψ_A, which affords an exact solution in slab geometry. In slab geometry, this method requires significantly more computer memory than most other schemes. Furthermore, the method does not appear to be extensible to other geometries (De Barros, et al., 1990). For these reasons, the SGF method is not considered for comparison to the EC scheme.

Multiple balance (MB) is a curve fitting scheme that uses the Boltzmann transport equation as the principle balance equation and auxiliary balance equations restricted to coupled half-cell domains. The MB method employs a Pade'(0,2) approximation to the exponential function. While this scheme has many desirable properties, it requires

computations over cell sub-domains, is limited to second order, and is not always positive when scattering is present in a given region (Morel, et al., 1990). Therefore, no performance comparison between multiple balance and exponential characteristic is made.

C. Characteristic Schemes

i. Step Characteristic Method

The step characteristic (SC) method uses a characteristic integration of the transport equation, which assumes the source function takes the form of a step value $s(x) = S_A$ in each cell. Thus, analytically integrating equation (2) along each characteristic direction (constant μ) yields solutions for the flux distribution and zeroth moment. This method is simple, positive, and conserves zeroth moments, but is difficult to generalize in multidimensional geometries. Also, this method demonstrates less than second order truncation error as cells become optically thin ($\sigma\Delta x \rightarrow 0$) (Lathrop, 1969:475).

ii. Linear Characteristic Method

The linear characteristic method is a logical extension of the step characteristic method. Instead of approximating the source function as a step value throughout a cell, this method uses a linear approximation of the source function in a characteristic integration of the transport equation:

$$s(x) = S_A P_0(x) + S_x P_1(x) \tag{13}$$

where the zeroth and first moments of the source function are S_A and S_x, respectively, and P_0 and P_1 are Legendre polynomials. These moments are initially obtained by integrating the product of true source function and the zeroth and first order Legendre polynomials. The true source function, as given in equation (3), is:

$$s(x) = c\sigma\phi(x) + s_{ext}(x) \tag{3}$$

The Legendre polynomials P_0 and P_1, defined so as to be orthogonal over the interval $0 \le x \le \Delta x$, are:

$$P_0(x) = 1 \tag{14}$$

$$P_1(x) = \left(\frac{2x}{\Delta x} - 1\right) \tag{15}$$

Integration is carried out using equations (3), (14), and (15) according to

$$S_A = \frac{1}{\Delta x}\int_0^{\Delta x} s(x)P_0(x)dx \tag{16}$$

$$S_x = \frac{3}{\Delta x}\int_0^{\Delta x} s(x)P_1(x)dx \tag{17}$$

These result in

$$S_A = c\sigma\phi_A + S_{extA} \tag{18}$$

$$S_x = c\sigma\phi_x + S_{extx} \tag{19}$$

Now substituting the linear approximation for $s(x)$ from equation (13) into the transport equation (equation (2)) and integrating analytically yields a solution for the angular flux distribution in the cell (where $\psi(0) = \psi_L$ is the left entering flux for a specified $\mu > 0$):

$$\psi(x) = \psi_L \exp\left[\frac{-\sigma x}{\mu}\right] + \frac{S_A - S_x}{\sigma}\left(1 - \exp\left[\frac{-\sigma x}{\mu}\right]\right) + \frac{-S_x 2\mu}{\sigma^2 \Delta x}\left(1 - \exp\left[\frac{-\sigma x}{\mu}\right]\right) + \frac{S_x 2x}{\sigma \Delta x} \tag{20}$$

Evaluating this equation at $x = \Delta x$ yields the right boundary angular flux ψ_R. Integration of equation (20) can be performed to obtain the zeroth and first angular flux moments using

$$\psi_A = \frac{1}{\Delta x}\int_0^{\Delta x} \psi(x) P_0(x) dx \tag{21}$$

$$\psi_x = \frac{3}{\Delta x}\int_0^{\Delta x} \psi(x) P_1(x) dx \tag{22}$$

The expressions for ϕ_A and ϕ_x in the zeroth and first source moments (equations (18) and (19)), are actually the angular flux moments integrated over all angles (in slab geometry) according to:

$$\phi_A = \int_{-1}^{+1} \psi_A(\mu) \frac{d\mu}{2} \tag{23}$$

$$\phi_x = \int_{-1}^{+1} \psi_x(\mu) \frac{d\mu}{2} \tag{24}$$

Naturally all angular flux integrations are performed numerically using the selected angular quadrature set under the discrete ordinates approximation (see equations (5) and (6)).

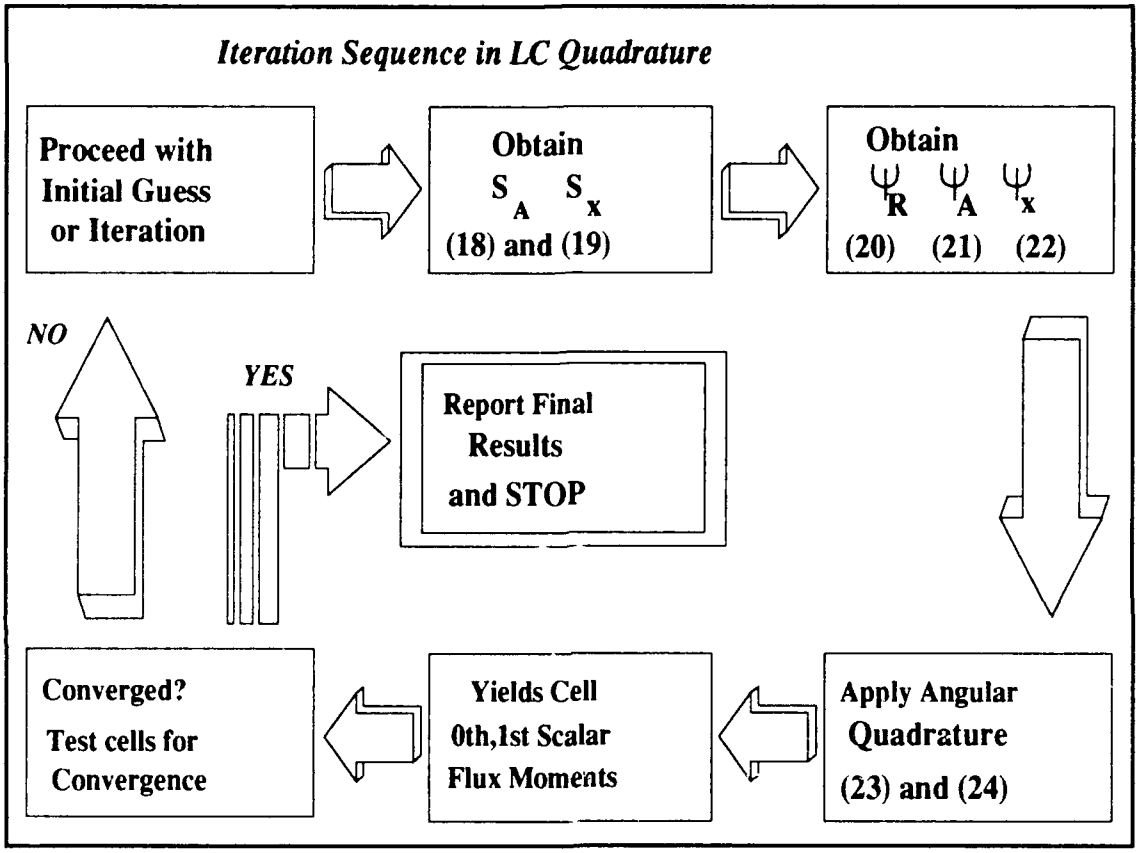

Figure 2. Iteration Sequence in Linear Characteristic Spatial Quadrature

A schematic of the progression from iteration to iteration in linear characteristic spatial quadrature is given in Figure 2. Since the source moments are either known from an initial guess or can be evaluated from a previous iteration, these yield new fluxes and flux moments, which provide new source moments for each cell. This process continues until convergence is achieved to a specified tolerance. Logically, this process is often referred to as fixed point iteration of the source-scattering term.

If $|S_x| > S_A$, which can occur when cells are optically thick, the linear source distribution equation (13) is negative at one end. To preserve positivity, a negative source fixup, also known as a rotational fixup, is required. This source fixup restricts the absolute value of S_x so that it is never larger than the zeroth source moment S_A. While this fixup insures positive sources (and hence positive fluxes), it violates first spatial moment conservation of the Boltzmann transport equation (and forces truncation error to increase). This, in turn, can cause numerical diffusion; particles are spatially shifted and can appear where they should not. The extent to which numerical diffusion occurs in LC depends on the impact of limiting the magnitude of S_x. In short, this source fixup can introduce problems analogous to those caused by negative flux fixups in the diamond difference method.

If the rotational fixup is never used, the linear characteristic method globally conserves the zeroth and first spatial moments of the Boltzmann transport equation. The zeroth spatial moment balance equation in slab geometry, the same balance equation used in the diamond difference approximation (equation (7)), is

$$\mu(\psi_R - \psi_L) + \sigma\Delta x \psi_A = S_A \Delta x \tag{25}$$

The first spatial moment balance equation is

$$3\mu(\psi_L - 2\psi_A + \psi_R) + \sigma\Delta x \psi_x = S_x \Delta x \tag{26}$$

These moment balances of the Boltzmann transport equation are easily derived by integrating equation (2) multiplied by the properly normalized zeroth and first order Legendre functions, equations (14) and (15), between the limits 0 and Δx. Over coarse meshes, the linear characteristic method is second order. In the limit of vanishingly thin cells, truncation of the LC method is at least third order (Alcouffe, et al., 1979:111-127).

In spite of potentially requiring a fixup in some cells (depending upon the problem), the linear characteristic method has produced better results than the diamond difference, linear discontinuous, exponential, and step characteristic methods in several test cases (Alcouffe, et al., 1979:126-127). In addition, it satisfies the correct diffusion limit (Larsen, 1983:90). Therefore, LC remains a worthy scheme with which to compare.

D. Adaptive Schemes

Mathews recently developed two methods, step adaptive (SA) and linear adaptive (LA) (Mathews, 1990:419-457). These are logical improvements to the step and linear characteristic methods, respectively. These new adaptive methods differ from earlier schemes in that source functions are strictly positive throughout the cell and require no rotational fixup. This is accomplished by partitioning the domain of each cell into smaller sub-domains. The source function is represented by different functions over each sub-domain. The operative "adaptive" for these schemes refers to the precise partitioning of the sub-domains in a way that forces the source representation to conserve 0th and 1st moments in a spatial cell. Hence, the source representation is determined by a moments matching process. Due to the intrinsic positivity of this method, no fixups are required, and zeroth and first moments of sources and flux distributions are always conserved, as are the balance equations (equations (25) and (26)). Note that these methods, while positive, are non-linear. This non-linearity has not been observed to interfere with convergence of the scattering iteration.

Not surprisingly, Mathews found SA and LA to be in excellent agreement with conventional methods in test problems with thin cells. However, when thick cells were

used, these adaptive schemes formidably outperformed the other methods. Truncation errors of third and fourth order are realized in the limit of very thin cells with SA and LA, respectively.

i. Step Adaptive Method

In the step adaptive method, a step function approximates the source function over a portion of the domain, and is truncated to zero in the remainder. The location of the truncation is dependent on the magnitudes of the zeroth and first moments and by the sign of the first moment.

For example, if $S_x \geq 0$, then

$$s(x) = 0 \qquad 0 \leq x \leq \rho\Delta x \qquad (27)$$

$$s(x) = \frac{S_A}{(1-\rho)} \qquad \rho\Delta x \leq x \leq \Delta x \qquad (28)$$

where

$$\rho = \frac{|S_x|}{(3S_A)} \qquad 0 \leq \rho \leq 1 \qquad (29)$$

If $S_x < 0$, then

$$s(x) = \frac{S_A}{(1-\rho)} \qquad 0 \leq x \leq (1-\rho)\Delta x \qquad (30)$$

$$s(x) = 0 \qquad (1-\rho)\Delta x \leq x \leq \Delta x \qquad (31)$$

ii. Linear Adaptive Method

The linear adaptive method works in a manner similar to SA, but is more accurate. If $|S_x| < S_A$, then $s(x)$ takes the form of equation (13), as in the LC method. However, if $S_x > S_A$, then $s(x)$ is set to zero from the left edge of the cell, up to a distance $(1-\tau)\Delta x$, where $\tau = 3(1-\rho)/2$. From this point, it rises linearly to a maximum at the right edge. A mirror image of this is used if $S_x < -(S_A)$. This insures the source function is continuous, positive, and piecewise linear in the cell; again, zeroth and first moments are conserved. In the limit of vanishingly thin cells in slab geometry, the linear adaptive method is equivalent to the linear characteristic method (Mathews, 1990:419-457).

E. Merits of an Exponential Characteristic Method

The true power of the step and linear adaptive schemes is in their ability to retain accuracy, positivity, and conservation with coarse meshes. This suggests the possibility of solving complex transport problems once thought too large or costly for conventional computer systems. One potential difficulty in using SA and LA involves adapting the source function to match moments over a sub-domain. If posed with a problem with strong and weak fluxes appearing at opposite boundaries of a given cell and $|S_x| > S_A$, the adapted source function is preferentially biased to de-emphasize the weak boundary flux, while the strong boundary flux is over-emphasized. An additional obstacle in SA and LA is simply the added complexity of using a piecewise source representation in sub-domains across a mesh cell. Depending on the problem being solved, use of sub-domains can require more floating point operations than might be necessary for a new method that uses a continuous source representation across an entire cell. (Use of sub-domains are also more difficult to treat in two or three space dimensions).

Still, any newly proposed cell-continuous source function must provide results that retain the features of accuracy, positivity, and conservation of at least zeroth and first moments (even over a coarse mesh). One source function that satisfies all these requirements is the exponential function

$$s(x) = a\exp[bx] \tag{32}$$

where a and b are constants to be chosen so as to conserve the zeroth and first spatial moments (i.e. to match the specified moments, S_A and S_x, from the outer iteration). This scheme is called the exponential characteristic (EC) method. While the constants in the source function in equation (32) are defined by moments matching similar to that used for SA and LA, the exponential characteristic method is not an adaptive scheme; it will not use sub-domains across a spatial cell. An added feature of this source function is its similarity to particle distributions in strongly absorbing media. Like the SA and LA schemes, the EC method is non-linear (as in schemes which use a fixup), and therefore does not preserve the linearity of the transport equation in the sense of the super-position of solutions.

III. Problem

The problem addressed here is the theoretical development, computer program implementation, validation, testing, and evaluation of a discrete ordinates exponential characteristic spatial quadrature scheme to numerically solve the Boltzmann transport equation. The EC method is evaluated based on a comparison of numerical results to results from other spatial quadrature schemes.

A. Scope

The "exponential characteristic" method is derived using a characteristic integration of the Boltzmann transport equation with an exponential function as the assumed form of the source distribution. Although the method derived here could be implemented more generally, for simplicity of programming, the implementation and testing performed here is restricted to the conditions stated in Table 2.

Table 2
Conditions Used in Testing the EC Method

1. Steady State
2. Mono-Energetic
3. Non-Multiplying
4. Isotropic Scattering and Sources in the Laboratory System
5. Slab Geometry

Validation and testing of the EC scheme is performed using test cases with multiple regions containing combinations of scattering and absorbing media. In each case, conservation of zeroth and first spatial moments is numerically verified using the balance

equations (equations (25) and (26)). In addition, an overall scalar balance equation (the steady state Boltzmann equation integrated over all angles) is also numerically verified. This overall balance equation is, in slab geometry,

$$J_x(\Delta x) - J_x(0) + \sigma \Delta x \phi_A = S_A \Delta x \tag{33}$$

where

$J_x(\Delta x)$ = net current at right cell boundary

$J_x(0)$ = net current at left cell boundary

σ = total macroscopic cross section

Δx = measured cell width

ϕ_A = scalar flux zeroth moment

S_A = source function zeroth moment

Note that the boundary currents $J_x(\Delta x)$ and $J_x(0)$ are given in slab geometry by

$$J_x(\Delta x) = \int_{-1}^{+1} \mu \psi(\Delta x, \mu) \frac{d\mu}{2} \approx \sum_{m=1}^{M} \mu_m w_m \psi_m(\Delta x) \tag{34}$$

and

$$J_x(0) = \int_{-1}^{+1} \mu \psi(0, \mu) \frac{d\mu}{2} \approx \sum_{m=1}^{M} \mu_m w_m \psi_m(0) \tag{35}$$

As before, all angular integrations are carried out using the discrete ordinates approximation.

Evaluation of the performance of the exponential characteristic method is, for any given test problem, achieved by comparing numerical results obtained from the EC method with results obtained from conventional spatial quadrature schemes in slab geometry. Comparisons are made among the schemes listed in Table 3.

Table 3
Spatial Quadrature Schemes Used in Test Problems

(EC) Exponential Characteristic (DD) Diamond Difference (DDF) Diamond Difference with Fixup (LD) Linear Discontinuous (LC) Linear Characteristic (SA) Step Adaptive (LA) Linear Adaptive

A relative convergence tolerance of 10^{-5} using Gauss-Legendre angular quadrature with eight direction cosines (S_8) is used to solve all problems. Only a numerical comparison of results is provided; no rigorous mathematical proof of the efficiency of the EC scheme versus other schemes is made.

Because equations (25), (26), and (33) are numerically verified for each quadrature scheme, including exponential characteristic, a check on all angular and scalar quantities is available. If numerical rounding errors or catastrophic cancellations have degraded solutions, this is immediately obvious from the inspection of the relative maximum differences in these balance equations. (Note that the first spatial moment balance equation (26) is not used for DD or DDF).

B. Error Norm and Error Treatment

For easy interpretation and evaluation of results, the error norm given by Mathews as

$$Err(approx) = \frac{|approx - exact|}{(|approx| - |exact|)/2} \tag{36}$$

is used throughout all studies to compare all cell edge fluxes, currents, and region moments, as appropriate. In addition, this same norm is used to compare balance equations (25), (26), and (33) for each quadrature scheme during computer execution, where the right and left hand sides of these equations are taken to be *approx* and *exact*, respectively. While many authors have used slightly different error norms, this norm is stable for zero or negative results, and approaches a conventional relative error estimate when the approximate value is close to the exact value (Mathews, 1990:446). By inspection, it is clear that the maximum value produced by this error norm is two.

Average and maximum observed errors are determined using error values computed from equation (36). An average pointwise relative error Q provides a measure of the accuracy of all computed values over an entire problem, and is given by

$$Q = \frac{1}{N} \sum_{n=1}^{N} Err(approx)_n \tag{37}$$

where N values for all fluxes, currents, etc, are computed for a given spatial quadrature method. The maximum observed error is the maximum relative error observed in any quantity for a given solution to a problem; therefore, it provides a "worst case" calculation error for a spatial quadrature method used to solve the problem.

As for the "exact" numerical solution from which errors are computed in equation (36), this is the reference solution obtained using the linear characteristic method with a

very fine spatial mesh (where further refinement of the mesh yielded no discernable change in the numerical answer). Unless otherwise stated, a relative tolerance of 10^{-5} is always used for convergence.

C. Computational Efficiency and Cost

In defining computational efficiency, an operational definition analogous to that stated by Alcouffe, et al., is used. This holds that one method is more computationally efficient than another if, for the same relative cost, the first method produces a more accurate solution (Alcouffe, et al., 1979:113). Here, the relative cost in solving any test case is determined by the size of the spatial mesh and the execution time required for each spatial quadrature method to reach a converged solution. Criteria for convergence is a comparison of scalar fluxes in all cells computed before and after the last iteration. If the relative change (using equation (36)) in these scalar fluxes is no larger than 10^{-5}, then the convergence test is satisfied.

The average error equation (37) is used to determine a mesh size ratio, or MSR. The MSR is defined as

$$MSR = \left[\frac{Cells(Q)_{XX}}{Cells(Q)_{EC}} \right] \tag{38}$$

where

$Cells(Q)_{XX}$ = number of cells required to achieve an average error Q using spatial quadrature scheme XX

$Cells(Q)_{EC}$ = number of cells required to achieve an average error Q using the EC scheme

XX = DD, DDF, LD, LC, SA, or LA as in Table 3.

Another useful ratio is the execution time ratio, or ETR. This is defined as

$$ETR = \left[\frac{ExecTime\,(Cells(Q))_{XX}}{ExecTime\,(Cells(Q))_{EC}} \right] \tag{39}$$

where

$ExecTime_{XX}$ = execution time required for method XX to achieve convergence using $Cells(Q)_{XX}$

$ExecTime_{EC}$ = execution time required for the EC method to achieve convergence using $Cells(Q)_{EC}$

XX = DD, DDF, LD, LC, SA, or LA as in Table 3.

Memory storage requirements are proportional to the number of mesh cells needed to achieve a specified error, and computational effort is related to the execution time necessary for convergence using the number of cells designated. Thus, while the average pointwise error Q provides a measure of overall solution accuracy, the *MSR* and *ETR* provide a comparison of memory storage requirements and computational effort, respectively, for any method XX against the memory storage and computational effort required for EC. The higher the value of either the *MSR* or *ETR* above unity, the greater the benefit from using the exponential characteristic scheme. A mesh cell ratio or execution time ratio less than unity for a given method indicates fewer cells or a shorter execution time, respectively, than for EC.

D. Source Iterations

Two approaches to the numerical iteration of the source function are used in the study of each test case. The first approach is a fixed point iteration of the source function, where scalar flux moments are updated between each iteration to provide zeroth

and first source moments (see Section II.C.ii). The second approach is the method of successive scatters. Iteration using successive scatters differs from fixed point iteration in that any external sources and incident currents present are used to inject a flood of neutrons into the system during the first iteration only; this yields the first flight flux. Subsequent iterations deal only with the scattered components of this first flight flux. Initially, the source term is

$$s(x) = s_{ext}(x) \tag{40}$$

For the second iteration and thereafter, the source term is

$$s(x) = c\sigma\phi_{Flight}(x) \tag{41}$$

where $\phi_{Flight}(x)$ is the scalar flux generated from scattering by the previous flight. As the scattering process continues, running sums of all fluxes and currents are accumulated until the scattered particle fluxes become negligible and convergence is achieved.

The treatment of the source function in successive scatters (given by equations (40) and (41)) differs from the fixed point iteration source term, which always includes components from both external sources and scatters, as in equation (3). Since all problems treated here are time independent, either method provides essentially the same solution. However, the method of successive scatters can be more accurate for a given mesh size. Because the external source term often dominates the source function in fixed point iteration, contributions from scattered radiation can be masked; this does not occur in successive scatters, as only scattering from a previous flight contributes to the source term (and thus to source moments for each iteration). This affects the accuracy of a solution, to an extent that depends on the spatial quadrature scheme used. This is demonstrated in a later section.

E. Programming

The exponential characteristic method is augmented into a slab geometry discrete ordinates computer code that has the capability of executing all other required spatial quadrature schemes. Programs were written in MicroSoft QuickBASIC 4.5 for execution on an IBM or compatible personal computer.

IV. Theoretical Development

The exponential characteristic method is derived in a manner very similar to that used for the linear characteristic method, except that the assumed form of the source function is an exponential. Again, a characteristic integration is performed over a single spatial cell, as in Figure 1.

A. Flux Distributions

Recalling the steady-state, one energy, transport equation in slab geometry (equation (2)):

$$\mu_m \frac{d\psi_m(x)}{dx} + \sigma_m \psi(x) = s(x) \tag{2}$$

Assume the isotropic source function for the exponential characteristic method has the form of equation (32):

$$s(x) = a\exp[bx] \tag{32}$$

Recall that a and b are to be determined by zeroth and first source moment matching. Integrating equation (2) by integrating factor along a direction cosine μ, and noting that at the left hand boundary, $\psi(0) = \psi_L$, results in

$$\psi(x) = \psi_L \exp\left[\frac{-\sigma x}{\mu}\right] + \int^{x} s(x')\exp\left[\frac{-\sigma(x-x')}{\mu}\right]\frac{dx'}{\mu} \tag{42}$$

Substituting equation (32) into equation (42) and carrying out the integration yields an expression for the angular flux distribution in a cell,

$$\psi(x) = \psi_L \exp\left[\frac{-\sigma x}{\mu}\right] + \left(\frac{a}{b\mu + \sigma}\right)\left(\exp[bx] - \exp\left[\frac{-\sigma x}{\mu}\right]\right) \tag{43}$$

Since ψ_R is defined as $\psi(\Delta x)$, evaluating equation (43) at $x = \Delta x$ yields the angular flux at the right cell boundary:

$$\psi_R = \psi_L \exp\left[\frac{-\sigma \Delta x}{\mu}\right] + \left(\frac{a}{b\mu + \sigma}\right)\left(\exp[b\Delta x] - \exp\left[\frac{-\sigma \Delta x}{\mu}\right]\right) \tag{44}$$

B. Flux Moments

As mentioned, a requirement imposed on the exponential characteristic scheme is that zeroth and first moments are to be globally conserved. The zeroth angular flux moment, recalling equation (21), is

$$\psi_A = \frac{1}{\Delta x}\int_0^{\Delta x} \psi(x) P_0(x) dx \tag{21}$$

Substituting equation (43) into this equation and integrating yields the simplified zeroth angular flux moment:

$$\begin{aligned}\psi_A = {} & \frac{\psi_L \mu}{\sigma \Delta x}\left(1 - \exp\left[\frac{-\sigma \Delta x}{\mu}\right]\right) + \left(\frac{-a}{b\sigma \Delta x}\right)(1 - \exp[b\Delta x]) \\ & + \left(\frac{-a\mu}{(b\mu + \sigma)\sigma \Delta x}\right)\left(\exp[b\Delta x] - \exp\left[\frac{-\sigma \Delta x}{\mu}\right]\right)\end{aligned} \tag{45}$$

Similarly, making use of equation (22) and again substituting equation (43) yields the angular flux first moment equation:

$$\psi_x = \left(\frac{-3\psi_L\mu}{\sigma\Delta x}\right)\left(1+\exp\left[\frac{-\sigma\Delta x}{\mu}\right]\right)+\left(\frac{6\psi_L\mu^2}{\sigma^2\Delta x^2}\right)\left(1-\exp\left[\frac{-\sigma\Delta x}{\mu}\right]\right)$$
$$+\left(\frac{3a}{b\sigma\Delta x}\right)(1+\exp[b\Delta x])+\left(\frac{-6a\mu}{b\sigma^2\Delta x^2}\right)(1-\exp[b\Delta x])$$
$$+\left(\frac{6a}{b^2\sigma\Delta x^2}\right)(1-\exp[b\Delta x])+\left(\frac{-3a\mu}{(b\mu+\sigma)\sigma\Delta x}\right)\left(\exp[b\Delta x]-\exp\left[\frac{-\sigma\Delta x}{\mu}\right]\right) \quad (46)$$
$$+\left(\frac{-6a\mu^2}{(b\mu+\sigma)\sigma^2\Delta x^2}\right)\left(\exp[b\Delta x]-\exp\left[\frac{-\sigma\Delta x}{\mu}\right]\right)$$

C. Source Moments

Maintaining the requirement that source moments be conserved, the zeroth source moment is defined as given in equation (16):

$$S_A = \frac{1}{\Delta x}\int_0^{\Delta x} s(x)P_0(x)dx \quad (16)$$

Substituting equation (32) into this equation yields a relationship between a, b, and S_A:

$$S_A = \frac{a}{b\Delta x}(\exp[b\Delta x]-1) \quad (47)$$

Again, the first source moment is defined as in equation (17),

$$S_x = \frac{3}{\Delta x}\int_0^{\Delta x} s(x)P_1(x)dx \quad (17)$$

and substituting equation (32) here yields a relationship between a, b, and S_x:

$$S_x = \left(\frac{3a}{b^2\Delta x^2}\right)[b\Delta x(\exp[b\Delta x]+1)+2(1-\exp[b\Delta x])] \quad (48)$$

To verify the expressions for the moments of all fluxes and sources, equations (44) to (48) were substituted into the balance equations (25) and (26) to yield mathematical identities.

D. Method

S_A and S_x are both known either as an initial guess or as evaluated from the previous iteration. The equations for S_A are as derived for the linear characteristic method. Recalling equation (18) (where c is given in equation (4)):

$$S_A = c\sigma\phi_A + S_{extA} \tag{18}$$

the zeroth scalar flux moment, using equations (21), (23), and the discrete ordinates approximation, is

$$\phi_A = \frac{1}{2}\sum_{m=1}^{M} w_m \psi_{A_m} \tag{49}$$

The external zeroth source moment is also defined as in the LC method:

$$S_{extA} = \frac{1}{\Delta x}\int_0^{\Delta x} s_{ext}(x) P_0(x)\,dx \tag{50}$$

Note that the value specified for an external source in a region for any test case is assumed to be the zeroth external source moment (average value of the source in the region), regardless of the spatial quadrature method. The first external source moment is, for the problems tested here, assumed to be zero. The corresponding equations defining S_x are, recalling equations (19), (22), and (24), again using the discrete ordinates approximation

$$S_x = c\sigma\phi_x + S_{extx} \tag{19}$$

$$\phi_x = \frac{1}{2}\sum_{m=1}^{M} w_m \Psi_{x_m} \tag{51}$$

$$S_{extx} = \frac{3}{\Delta x}\int_0^{\Delta x} s_{ext}(x) P_1(x)\, dx \tag{52}$$

The Legendre functions are as in equations (14) and (15). A schematic of the progression from iteration to iteration in exponential characteristic spatial quadrature is given in Figure 3.

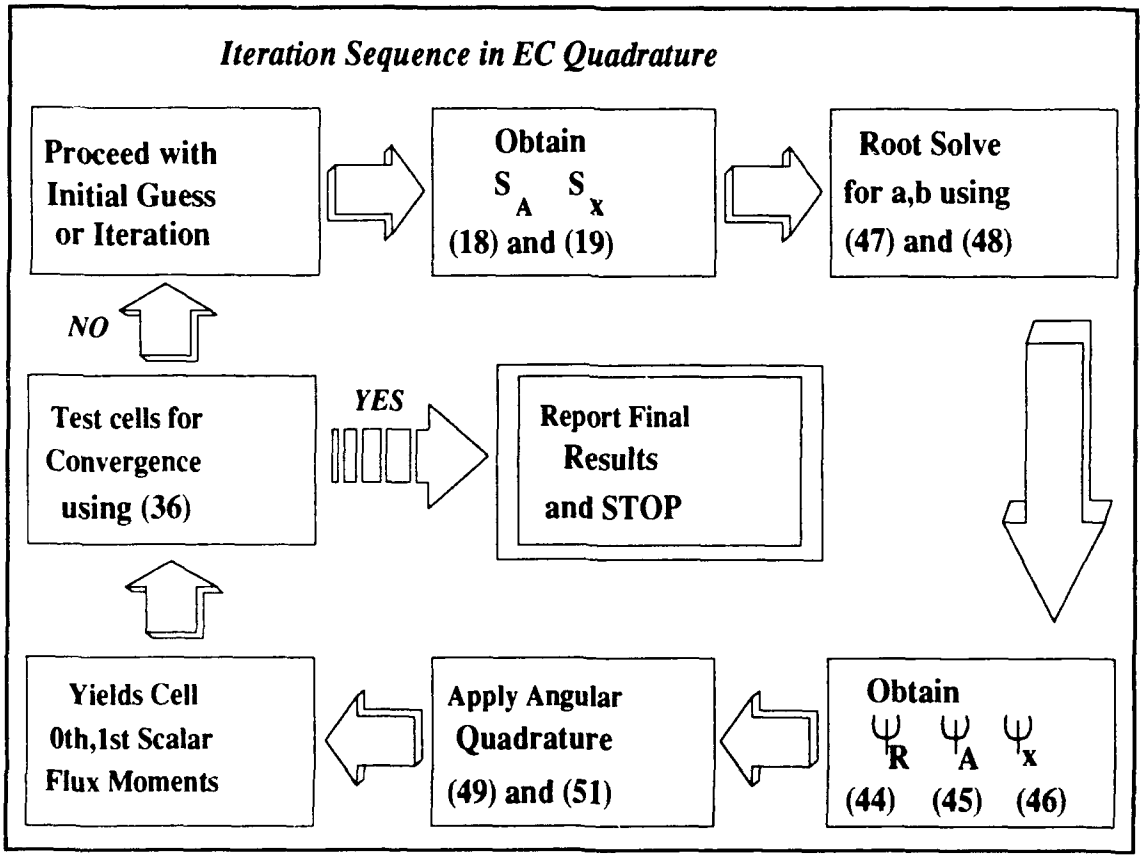

Figure 3. Iteration Sequence in Exponential Characteristic Spatial Quadrature

In each iteration, S_A and S_x are computed from equations (18) and (19), and coefficients a and b are determined by root solving the source moment relationships given in equations (47) and (48). Values for flux moments and fluxes are then computed using equations (44), (45), and (46). This process continues over all directions and cells until a converged solution is reached. Since the source function is never negative, no fixups are required, and the EC method globally conserves zeroth and first spatial moments. Because of this, the balance equations given in equations (25) and (26) are satisfied in each cell for each discrete ordinate. In view of this, the overall balance, equation (33), must also be satisfied. This is true unless numerical rounding errors, catastrophic cancellations (overflows/underflows), or use of numerical formulations that are ill-conditioned compromise the solutions.

E. Conversion to Computable Forms

The equations derived for the exponential characteristic method are not, in all cases, numerically stable. For example, consider equation (44), the angular flux on the right boundary.

$$\psi_R = \psi_L \exp\left[\frac{-\sigma\Delta x}{\mu}\right] + \left(\frac{a}{b\mu+\sigma}\right)\left(\exp[b\Delta x] - \exp\left[\frac{-\sigma\Delta x}{\mu}\right]\right) \qquad (44)$$

When the constant b is zero or negative, the second term in equation (44) can become unstable. By inspection, similar difficulties are evident in the flux and source moment expressions presented in the previous section. The conversion of each equation to a compact, computable form is necessary to insure that any iteration procedure is sound.

i. Walters Functions

Walters introduced a recursive set of exponential functions that implicitly occur often in solutions to the Boltzmann transport equation (Walters, 1981:115 and 1986:192-196). The first three of these expressions, used by Mathews with a lower case "p" (to avoid confusion with Legendre polynomials), with $z \geq 0$ are

$$p_0(z) = \frac{1 - \exp(-z)}{z} \tag{53}$$

$$p_1(z) = \frac{1 - p_0(z)}{z} \tag{54}$$

$$p_2(z) = \frac{1 - 2p_1(z)}{z} \tag{55}$$

Terms where $n \geq 1$ can be summarized by the general Walters function forward iteration formula

$$p_n(z) = \frac{1 - n p_{n-1}(z)}{z} \tag{56}$$

Taking the limit as the argument of equation (56) approaches zero yields a general expression to be used when $z = 0$:

$$p_n(0) = \frac{1}{n+1} \tag{57}$$

Equations (53), (54), and (55) are plotted in Figure 4, along with $1/z$ and $\exp(-z)$, with $z \geq 0$, for comparison.

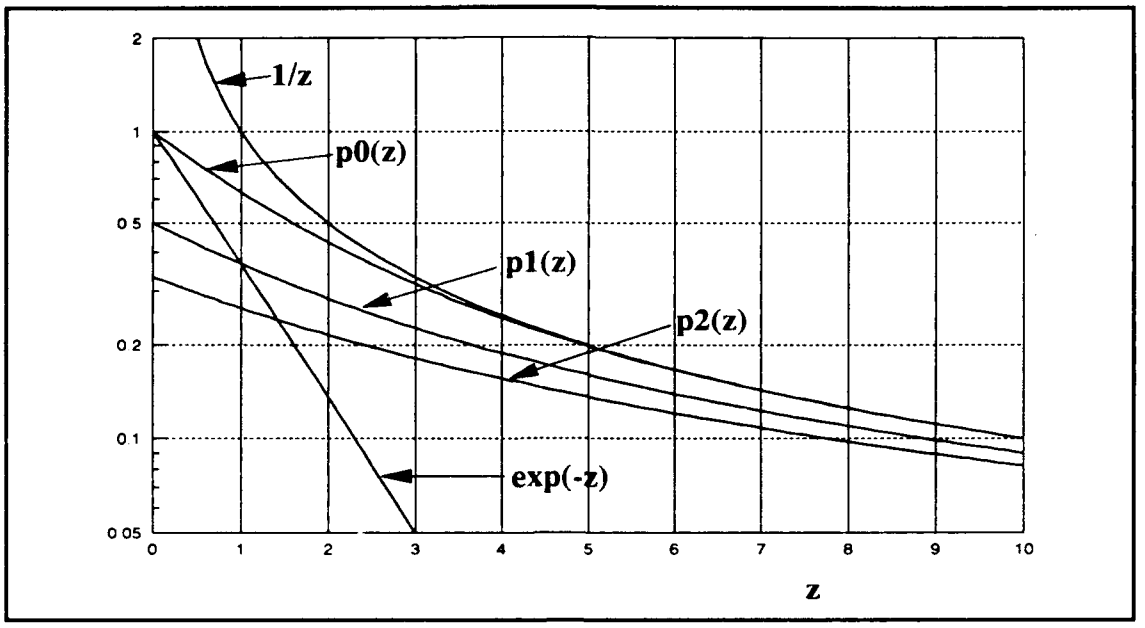

Figure 4. Functions 1/z, exp(-z), and Walters Functions $p_0(z)$, $p_1(z)$, and $p_2(z)$

As described by Mathews, when $z > n$, equation (56) can accurately be used to successively compute the next higher term, up to the level required in a computation. Again, this is known as forward iteration. If $z < n$, the forward iteration process becomes increasingly less accurate, losing precision with each iteration. To overcome this, Mathews suggests that backward iteration should be performed by rearranging equation (56), so that

$$p_{n-1}(z) = \frac{1 - z p_n(z)}{n} \tag{58}$$

which is the form for backward iteration. Remarkably, if n is commenced at a high enough value, any initial guess in [0,1] can be used to begin the backward iteration procedure, whereupon an exact solution for the Walters function can result. The worse the initial guess, the higher the order n must be (Mathews, 1990:431-432). The error of

any initial guess p_{n_0} can be no greater than the value of the initial guess if we use $1/(2(n_o+1))$ as the initial guess (since $1/(n_o+1) \geq p_{n_o}(\varepsilon) \geq 0$ for $\varepsilon > 0$). For a Walters backward iteration scheme that begins at n_0, the absolute error in the nth Walters function using backward iteration is

$$AbsErr\,p_n(z) = \frac{p_{n_0}(z)\,z^{(n_0-n)}n!}{n_0!} \tag{59}$$

and the relative error is

$$Err\,p_n(z) = \frac{AbsErr\,p_n(z)}{Exact\,p_n(z)} \tag{60}$$

A relative error of less than 2.0×10^{-17} can be realized for $n = 0 to 3$ in backward iteration if $p_{27}(z) = 1/56$ is used as an initial guess for $n_0 = 27$.

For all transport calculations treated here, the highest order Walters functions required for any of the candidate methods in Table 3 is $p_3(z)$. If the truncated integer portion of z minus one is less than or equal to three, backward iteration is implemented until the order of the Walters function n is equal to the truncated integer part of z. Forward iteration is used thereafter or when the truncated integer part of z minus one exceeds three. For example, suppose $z = 2.5$. The truncated integer portion of z less one is 1 (the truncated integer portion of 2.5 is 2). Since $1 < 3$, then backward iteration is used to compute $p_3(2.5)$ and $p_2(2.5)$, and forward iteration is used to compute $p_0(2.5)$ and $p_1(2.5)$. ("Walters" functions are used for all the methods except DD and LD. The linear discontinuous scheme is algebraically equivalent to the linear nodal (and hence linear characteristic) scheme when a Pade' (1,2) approximation $\exp(-x) \approx (6-2x)/(6+x(x+4))$ is used in linear nodal; this Pade' approximation is implemented when LD is executed, and therefore use of Walters functions is not necessary).

If negative arguments are placed into the Walters functions, values begin to increase exponentially, and vital digits can be lost in successive calculations of large positive numbers. To avoid this, the original Walters functions are easily transformed to better accommodate negative arguments using combinations of the original Walters functions to any order desired. The transformations for equations (53), (54), and (55) for $n = 0 to 2$ are (for $z > 0$):

$$p_0(-z) = [p_0(z)]\exp(z) \tag{61}$$

$$p_1(-z) = [p_0(z) - p_1(z)]\exp(z) \tag{62}$$

$$p_2(-z) = [p_0(z) - 2p_1(z) + p_2(z)]\exp(z) \tag{63}$$

Using positive arguments and the proper iteration approach (forward or backward), stable values can be achieved. Care must be exercised in using these formulations; subtraction of nearly equal numbers can occur in some instances, resulting in a loss of numerical precision. Formulation of (61), (62), and (63) in truncated Maclaurin expansions of suitable order is recommended in such cases. These expansions were not required for problems tested here.

ii. Flux Equations

Implementing the Walters functions into equations (44) to (48) results in a very compact set of equations. The right boundary angular flux in equation (44) is equivalent to

$$\psi_R = \psi_L \exp[-\varepsilon] + \frac{a\Delta x}{\mu} p_0(\beta - \varepsilon)\exp[-\varepsilon] \tag{64}$$

where ε is the optical thickness for a given μ as in equation (12), and $\beta=(-b\Delta x)$. The coefficients a and b are from the assumed source function (equation (32)). Similarly, the zeroth and first moments of the angular flux are

$$\psi_A=\psi'_L p_0(\varepsilon)+\frac{a\Delta x}{\mu\varepsilon}\left[p_0(\beta)-p_0(\beta-\varepsilon)\exp[-\varepsilon]\right] \tag{65}$$

and

$$\psi_x=3\psi_L\varepsilon[p_2(\varepsilon)-p_1(\varepsilon)]+\frac{3a\Delta x\beta}{\mu\varepsilon}[p_2(\beta)-p_1(\beta)]$$
$$+\frac{3a\Delta x}{\mu\varepsilon^2}[2p_0(\beta)-[\varepsilon+2]p_0(\beta-\varepsilon)\exp[-\varepsilon]] \tag{66}$$

Since negative and positive argument forms for Walters functions are available using both forward and backward iteration schemes (as applicable), no specific treatment of terms using $(\beta-\varepsilon)$ are typically required. Also, although $z(p_2(z)-p_1(z))$ is equivalent to $(p_0(z)-2p_1(z))$, the first formulation is used in equation (66) (and wherever applicable). This is because as $z\rightarrow 0$, $(p_2(z)-p_1(z))$ approaches -1/6, while the other term is ill-conditioned and goes to zero. Further, although division by ε seems undesirable, these are the most practical forms for these equations. In the event that optical thicknesses become extremely thin to the point of causing catastrophic cancellations or overflows, Maclaurin expansions of terms involving ε could be implemented. However, numerical testing of equations (64), (65), and (66) as presented with 0.003 revealed no difficulties. Moreover, since the exponential characteristic scheme is designed to achieve high accuracy using coarse cells, use of an extremely fine mesh is unnecessary.

iii. Source Equations and Root Solving

In order for the exponential characteristic method to yield a solution to the Boltzmann transport equation, very accurate values for the coefficients a and b are required. As mentioned earlier, equations (47) and (48) relate S_A and S_x to the source functions (equations (18) and (19)). Since S_A and S_x are known, root solving can be performed using equations (47) and (48) to determine a and b for each spatial cell in a problem. Again making use of the Walters functions, the following equations result from a manipulation of equations (47) and (48):

$$S_A = a\,p_0(\beta) \tag{67}$$

$$S_x = [3a\beta(p_2(\beta) - p_1(\beta))] \tag{68}$$

Equation (67) can be solved directly for a to give

$$a = \frac{S_A}{p_0(\beta)} \tag{69}$$

Using this and further manipulating equation (68) results in an equation that requires root solving for β, which indirectly yields b, since $\beta = -b\Delta x$:

$$\frac{p_1(\beta)}{p_0(\beta)} = r_0 \tag{70}$$

where

$$r_0 = \frac{(1-\rho_x)}{2} \tag{71}$$

and

$$\rho_x = \frac{S_x}{3S_A} \tag{72}$$

Note that no absolute values are specified for equation (72), which differs from equation (29) in the adaptive methods (Mathews, 1990:419-457). Equation (70) can be expressed in an expanded form using exponentials:

$$\left[1 - \frac{1}{\beta} + \frac{1}{\exp[\beta] - 1} - r_0\right] = 0 \tag{73}$$

A plot of the function $p_1(\beta)/p_0(\beta)$ is given in Figure 5.

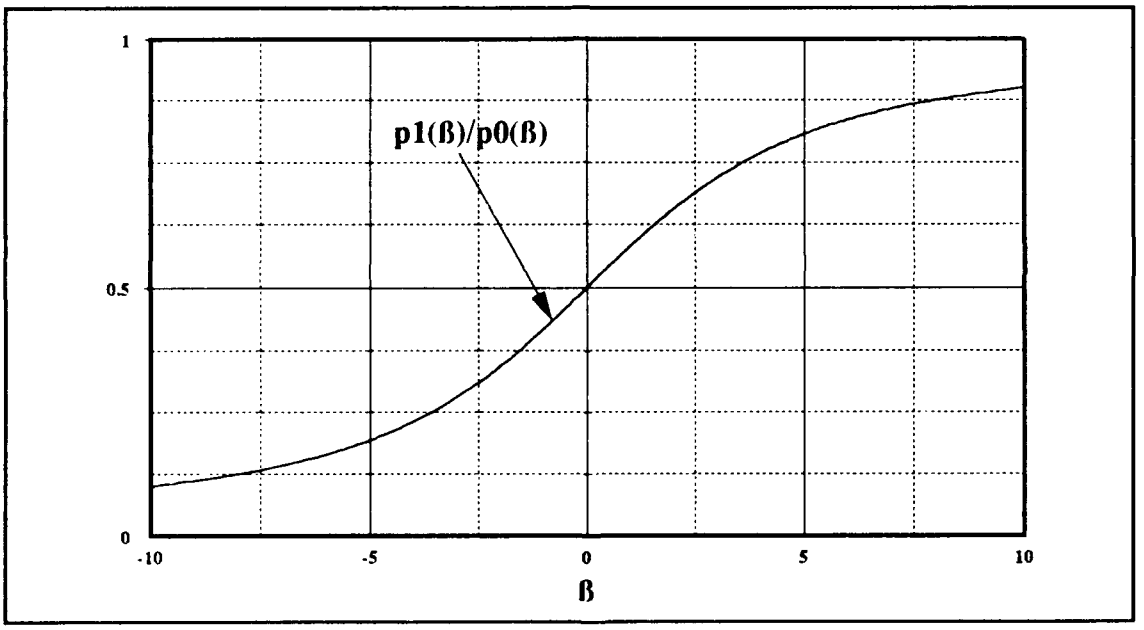

Figure 5. Ratio of $p_1(\beta)$ to $p_0(\beta)$ Walters Functions

An efficient root solving scheme is required to make the exponential characteristic scheme competitive with other discrete ordinates methods. Newton's method is preferred to determine β for each cell, as it has a second order of convergence. One difficulty in using Newton's method is that it requires a reasonable initial guess for the root to

guarantee convergence. Upon inspection of Figure 5, note that $p_1(\beta)/p_0(\beta)$ is comparable to an arctan(β) function. Normalizing the arctangent function so that it carries the proper limits results in

$$\frac{p_1(\beta)}{p_0(\beta)} = \left(\frac{1}{2} + \frac{1}{\pi}\arctan\left[g(\beta)\right]\right) \tag{74}$$

where $g(\beta)$ is some function which adjusts the curvature of the equation to yield an exact result. Solving for $g(\beta)$ in equation (74) and expanding the result in a Maclaurin series to first order yields

$$g(\beta) \approx \left(\frac{\pi\beta}{12}\right) \tag{75}$$

Placing this approximation for $g(\beta)$ back into equation (74) yields a good approximation for $p_1(\beta)/p_0(\beta)$:

$$\frac{p_1(\beta)}{p_0(\beta)} \approx \left(\frac{1}{2} + \frac{1}{\pi}\arctan\left[\frac{\pi\beta}{12}\right]\right) \tag{76}$$

Replacing $p_1(\beta)/p_0(\beta)$ in equation (70) with equation (76) and solving the result for β yields:

$$\beta \approx \frac{12}{\pi}\tan\left[\left(r_0 - \frac{1}{2}\right)\pi\right] \tag{77}$$

Note that adding higher order terms in the Maclaurin expansion for $g(\beta)$ forces one to solve for a root of a quadratic, which defeats the purpose of finding a simple first guess for β. Equation (77) provides an excellent first guess to the root β when $0.23 < r_0 < 0.77$,

with a worst case absolute error of ≈ 0.3 at either endpoint, becoming nearly exact as $r_0 \to \frac{1}{2}$, where $\beta \to 0$. A hyperbolic tangent can also be constructed to approximate a first guess for the root; however, the formulation using a tangent function is more accurate.

The tangent approximation (equation (77)) for β can be used for the entire range of r_0, which asymptotically approaches one for positive β and zero for negative β. Thus, β extends to infinity in both the positive and negative directions. A first guess better than that from equation (77) when $r_0 > 0.77$ is obtained by observing the limits of the actual equation (73) for large positive values of β. When $r_0 \geq 0.77$, then

$$\beta \approx \frac{1}{1 - r_0} \tag{78}$$

Similarly, when $r_0 \leq 0.23$,

$$\beta \approx -\frac{1}{r_0} \tag{79}$$

These approximations are also very good, providing a root to within ≈ 0.3 where they take over from equation (77), becoming nearly exact as r_0 approaches zero or one, respectively.

Using equations (77), (78), or (79) as appropriate for an initial guess of β, the following equations for Newton's method yield roots within 5 iterations at an absolute tolerance of 10^{-15} (determined by numerical testing).

$$\beta_{n+1} = [\beta_n - \frac{G(\beta_n, r_0)}{G'(\beta_n, r_0)}] \tag{80}$$

where

$$G(\beta_n, r_0) = \left[1 - \frac{1}{\beta_n} + \frac{1}{(\exp[\beta_n] - 1)} - r_0 \right] \tag{81}$$

and

$$G'(\beta_n, r_0) = \left[\frac{1}{\beta_n^2} - \frac{1}{(\exp[\beta_n] - 1)} - \frac{1}{(\exp[\beta_n] - 1)^2} \right] \tag{82}$$

As of this writing, the success of the exponential characteristic scheme in slab geometry (as is shown in later sections) has prompted Minor to begin development of the EC method in two dimensional rectilinear geometry. Since the task of root solving in each cell constitutes the bulk of computational effort in the EC scheme, any substantial improvement in root solving can significantly reduce the execution time required for EC. Minor has investigated alternative root solving schemes to improve computational efficiency. Using equation (71) and solving equation (70) for ρ_x and expanding the numerator and denominator of the result in a Maclaurin series, Minor found (in preliminary studies) that the number of iterations required for root solving might be reduced by at least one iteration. This requires use of the Maclaurin expansion over defined sub-domains wherein varying numbers of higher order terms are carried to obtain double precision accuracy. Minor is also investigating the utility of a table-interpolation of Figure 5 to obtain a first guess for root solving (Minor, 1991).

F. EC Source Function Behavior

In order to cast the distribution functions and moments into Walters functions, the quantity $\beta = -b\Delta x$ was introduced, where b is constant from the source function (equation (32)). When β is negative, $b > 0$ and the first source moment S_x is positive. Conversely, a positive value of β indicates S_x is negative. It is easily seen that in a

symmetrical problem, both positive and negative values of β are encountered, emphasizing the need to treat Walters functions using negative arguments (see Section IV.E.i.).

Mathematically, the exponential characteristic scheme is robust due to the unique behavior of the source function $s(x) = a[\exp(bx)]$. This function appears to be a natural choice if faced with solving deep penetration transport problems. If b is identically zero, then all equations describing flux distributions and moments default to the step characteristic method. This is useful in problems that result in absolutely flat flux profiles. In the case where $\pm b$ is very small but non-zero, as in a region with a virtually flat flux profile, the source function can be approximated by the truncated Maclaurin expansion

$$s(x) \approx a(1 + bx) \tag{83}$$

Using this approximation in place of equation (32) to derive the flux distributions and moments in equations (42) to (48), the exponential characteristic method simplifies into an algebraic equivalent of the linear characteristic method.

V. Program Development and Validation

Program development, as discussed here, consisted of three phases: standardization, optimization, and computer implementation of the exponential characteristic spatial quadrature method. Standardization dealt with adapting existing codes using candidate comparison methods (DD, DDF, LD, LC, SA, and LA) into one complete code, so that all quantities could be computed using standard variables and methods. To optimize the single multi-method code, several steps were taken, including the installation of timing sequences, initializations, and moment and overall balance checks, as well as the streamlining of data reporting and storage. Finally, the exponential characteristic method, using computable forms for equations derived in the previous section, was implemented into the multi-method code. As for program validation, all methods were continuously validated and checked for accuracy following each major code modification.

A. Standardization and Optimization

Several codes were supplied by LCDR K. Mathews, Ph.D. for solving slab geometry discrete ordinates problems. Separate codes using diamond difference (with a fixup option), linear discontinuous, step adaptive, and linear adaptive spatial quadratures were supplied. A linear characteristic routine was constructed by modifying the linear adaptive model. Because each code was slightly different in the way values were computed, every effort was made to standardize all calculations into a single, multi-method code. A standardized code was successfully implemented and allows the computational effort for any spatial quadrature scheme to be determined directly (for a specific problem) from the execution time required to achieve a converged solution. All

execution times were computed based on real clock inquiries from the computer operating system. (To avoid single system multi-tasking timing conflicts, no transport solutions were obtained under the MicroSoft Windows 3.0 PC environment).

The boundary conditions treatable in this multi-method code include a choice of a vacuum, a symmetric albedo, or a grey albedo at each slab face. Types of incident currents available, also at each slab face, are a Lambertian, an isotropic surface source, or a collimated beam (at any angle of incidence to the slab). These boundary conditions and incident currents are discussed further in Appendix B. All iterations begin at the left slab edge with positive angles and progress from left to right, then from right to left with negative angles. Fluxes and currents are all initialized to zero as each different spatial scheme is applied to a problem. The balance equations (equations (25) and (26)) are verified for each cell during each iteration. The left and right sides of these equations are compared, and the resulting maximum differences are reported. Upon convergence to a specified relative tolerance (10^{-5}), angular quadratures are performed and scalar fluxes and currents are globally computed. Region values are then calculated by folding together cell values. The overall balance equation (equation (33)) is verified in a manner similar to that for equations (25) and (26).

All input files for any problem can be set up in a standard format specified in the MAKEIN1D input code. Due to unique differences involved in treating the source function using fixed point and successive scatter methods, separate multi-method codes are used for each technique. The name assigned to the fixed-point multi-method code is SN1D-ALL, while the name assigned to the successive scatters version is SS1D-ALL. Full featured output options are available, including ASCII data files which facilitate

plotting of solutions. In addition, a separate error and order of convergence processing algorithm, PROCSNO2, was developed to deal with the results from a large collection of transport solutions for any given problem.

B. Implementation of EC Spatial Quadrature

Following all of the modifications required for a smooth, multi-method, standardized code, only one step was required to implement exponential characteristic spatial quadrature. This step consisted of developing the subroutine which computed the angular fluxes, based on source moments, for each mesh cell.

Naturally, the subroutine for the exponential characteristic method had to include code for root solving to determine the coefficients a and b in equation (32). Following an initial guess for β using either equations (77), (78), or (79), root solving using equations (80), (81), and (82) was performed. Performance studies were conducted to determine the optimum relative tolerance used in the Newton iteration loop. A relative root tolerance of 10^{-5} was selected. This value provided a root that was accurate enough for problem convergence to 10^{-5}, yet allowed roots to be found rapidly. In addition, flux moments computed from roots calculated using this tolerance satisfied the balance equations (See equations (25), (26), and (33)) as well as the other methods. Typically, between two and five iterations were required to find a root. Decreasing the root solving tolerance below 10^{-5} causes EC to require increased execution time. As shown in test problems presented in subsequent sections, use of a relative tolerance of 10^{-5} offered excellent relative performance.

Following the determination of the root $\beta = -b\Delta x$, Walters functions using β and ε (the angular optical thickness) were calculated. If needed, the negative arguments for the

Walters functions were treated using equations (61), (62), and (63), and all required fluxes and flux moments were calculated using equations (64), (65), and (66). No other functions were necessary inside the exponential characteristic spatial quadrature step. The subroutine which performs these tasks is "StepEC" in the source code for SN1D-ALL in Appendix D.

C. Validation

The multi-method slab geometry codes were verified by comparing solutions of conventional methods with independent versions of the originally supplied codes for DD, LD, SA, and LA. An additional validation was performed using one independent slab geometry code employing the DD scheme. Problems used to verify all solutions consisted of a selection of optically thin single and multiple region problems (with various scattering and absorption properties). All methods yielded essentially identical numerical solutions to these problems. Solutions to these validation problems using the exponential characteristic method agreed with those derived using conventional methods.

VI. Testing and Evaluation

While many slab geometry transport problems were solved during the development of the exponential characteristic scheme, only the results from a few select test cases are presented. Clearly, there are an infinite number of scenarios where the performance of the EC method versus other spatial quadrature methods might be investigated. However, problems put forth here best illustrate the strengths and weaknesses of the exponential characteristic method discovered to date. Each test problem was solved using both fixed point and successive scatter iterations of the source function with various mesh sizes corresponding to a range of optical thicknesses $\sigma\Delta x$. Limiting assumptions, comparisons to other methods, error analyses, and computational costs and efficiencies are treated as described in Section III.

A. Test Case 1: Thick Absorber

To initially challenge the exponential characteristic method and evaluate its utility, a single region with an optical thickness of sixteen (used by Mathews in his evaluation of the linear adaptive method) was used as the first test case (Mathews, 1990: 444-445). A schematic of this problem is presented in Figure 6.

The source on the left boundary (an ideal surface of isotropic emitters) has an angular flux distribution inverse to μ, corresponding to large fluxes at steep incident angles (See Appendix B). Augmented by a small amount of scatter, the scalar flux on the left boundary for the reference solution is more than 3.0. The region average scalar flux, as expected, is reduced to nearly 1/45th of this value, reaching a value of $\approx 10^{-8}$ at the right boundary. The reference solution was computed with the linear characteristic method using 256 cells over the single region. All methods performed well at the left

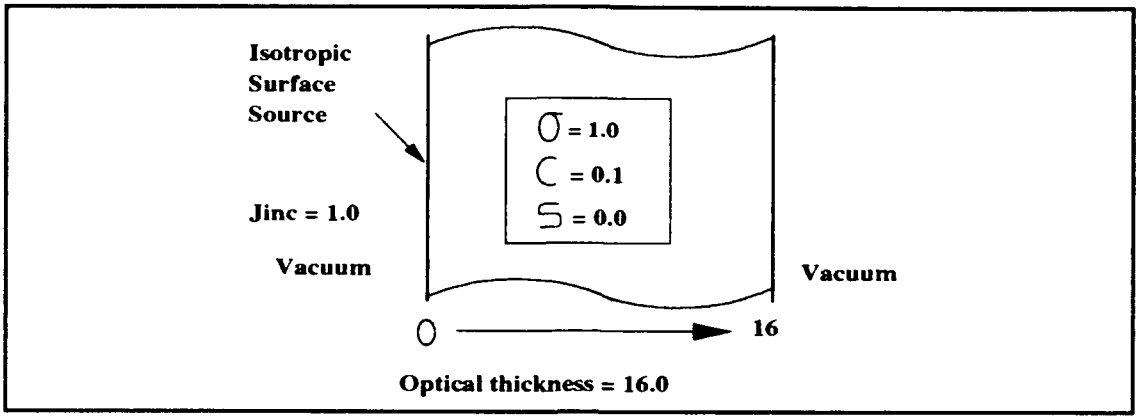

Figure 6. Schematic for Test Case 1

boundary where current enters the slab. However, of real interest is how accurately the various methods (listed in Table 3) computed the region average and right boundary (x = 16) scalar fluxes. Error norms for each spatial quadrature method for the region average scalar flux are presented in Figure 7. This figure clearly shows that the exponential characteristic (EC) method outperformed all of the others, achieving results for average flux within less than 0.2 percent of the reference solution using an optical thickness of 4 (corresponding to only 4 mesh cells). Also note that EC performed nearly as well using only a single sixteen mean free path cell.

The error norms for the right boundary flux are presented in Figure 8, where again, the exponential characteristic method yielded much more accurate results using thick cells. Observe in Figure 8 that using an optical thickness $\sigma\Delta x = 4$, the error of the EC right boundary flux was less than 2 percent, while for LA, the nearest competitor, the error was greater than 10 percent. On the coarsest mesh, EC performed slightly better than LA. These results are typical of those found for currents and first flux moments as

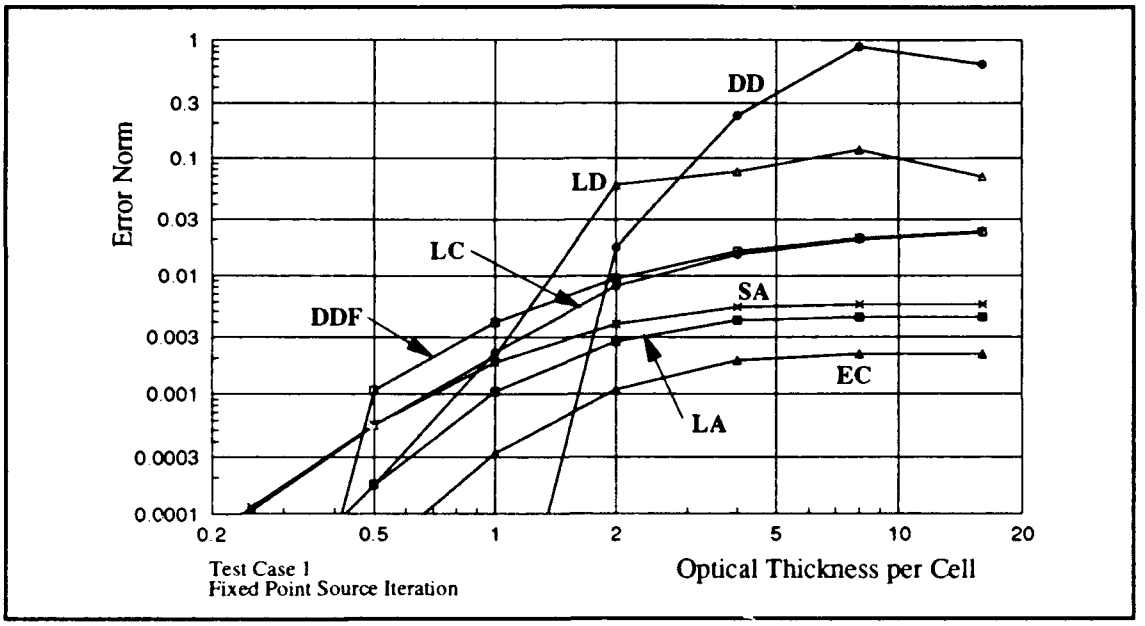

Figure 7. Test Case 1 Error in Region Average Scalar Flux

well. Note that while the diamond difference (DD) scheme converged rapidly as $\sigma\Delta x \rightarrow 1$ for the region average flux in Figure 7, it was severely in error at this same optical thickness at the right boundary, as shown in Figure 8.

In terms of average pointwise error Q, computed using equation (37), the EC scheme was almost four times more accurate than its nearest competitor, LA. Also, as can be discerned from Figures 7 and 8, the EC method maintained a bounded error using optically thick cells in this problem. As the mesh became highly refined, EC showed at least 4th order convergence. (Convergence appeared to be 5th order in the limit of very thin cells; this was not confirmed due to limitations in root solving, which required a shift to LC when using extremely fine cells, as discussed in the next section).

Mesh size ratio (MSR), discussed in Section III.C, is useful in illustrating a normalized mesh size relative to the number of cells used by the exponential

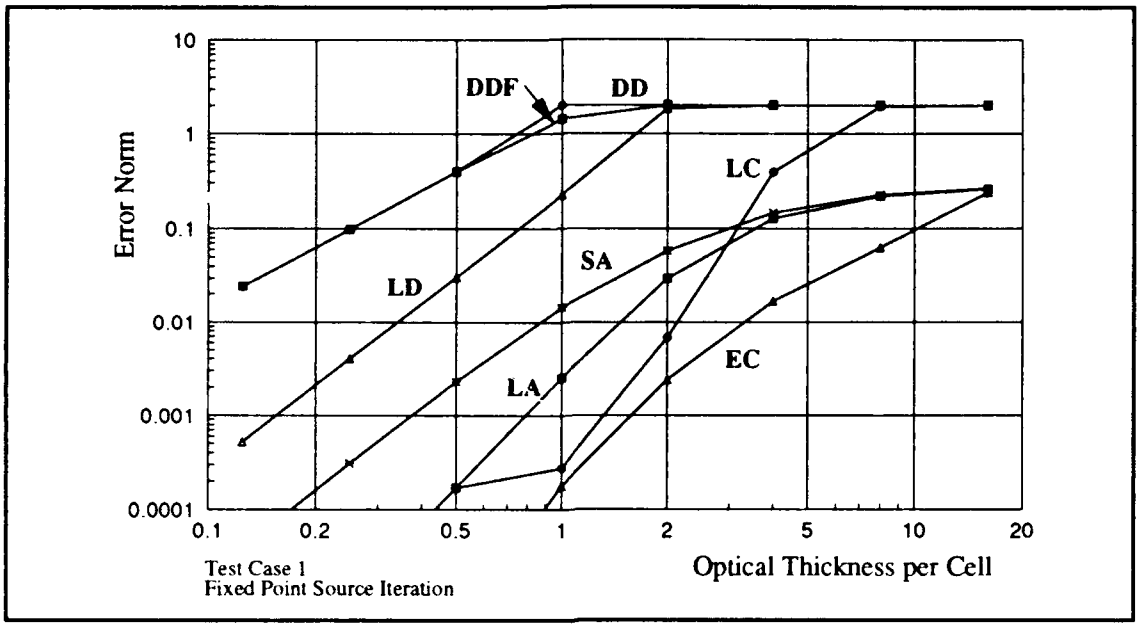

Figure 8. Test Case 1 Error in Right Boundary Scalar Flux

characteristic scheme to achieve an average pointwise error Q. Figure 9 is a graph depicting the relative MSR for this single thick absorber problem. (MSR values were computed using linear interpolation of average pointwise error results over optical thicknesses typical of those shown in Figures 7 and 8). Observe in Figure 9 that when 4 mesh cells were used for EC quadrature (yielding an error Q less than 1.5 percent), the MSR for linear adaptive quadrature was 2.5. This means that for LA to achieve a average pointwise error less than 1.5 percent, two and one half times the number of cells used by EC were necessary (10 cells).

Figure 9 further demonstrates that significantly more cells are required by the other methods for an equivalent average error; DD and DDF would require such an abundance of cells to obtain a calculation equivalent to EC that they offered no competition. Note from the figure that both LD and SA appeared to level off with MSRs of nearly 5 and 3,

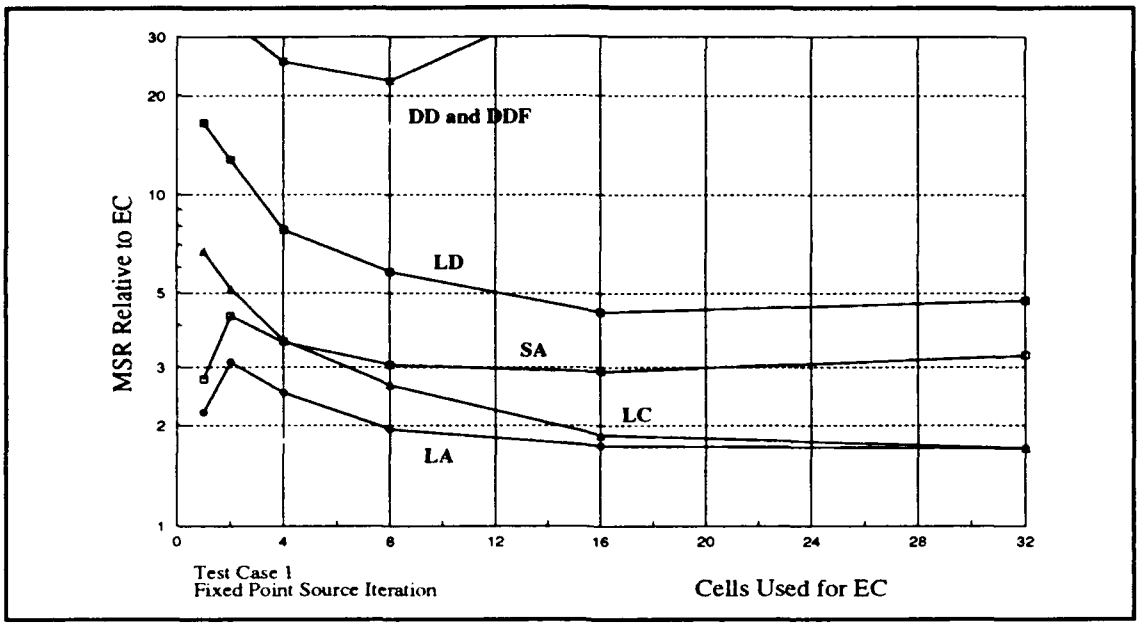

Figure 9. Test Case 1 Mesh Size Ratio

respectively, while LC and LA merged to an MSR value of 1.7 as the mesh was refined. Obviously, for this single thick absorber, EC is superior. In comparing execution times required to reach an equivalent error Q, the EC method offered an advantage when up to five cells were used (with an error Q of one percent). However, execution times for single region absorption problems were so low that actual comparisons are made only in later problems with at least two regions.

Solutions for this problem using successive scatters of the source function yielded similar results. Overall, errors computed for successive scatters were 10 percent lower for exponential characteristic quadrature, and remained mostly unchanged for all of the other methods. Apparently, EC benefits from tracking each scattered flight of neutrons in this thick absorber, possibly because it more accurately represents the source moment behavior of each flight with the exponential source function. In any event, exponential

characteristic has proven to be a very robust scheme in test case 1.

B. Test Case 2: Moderate Scatter and Source

Because the exponential characteristic method showed much promise in the first one region absorber problem, two different regions were specified for the second test case. A schematic of this problem is given in Figure 10. The first region is a sixteen mean free path thick absorber with moderate scattering and is coupled to a second equally thick absorbing region containing a source. The left boundary is a vacuum, and the right boundary is an isotropic surface source with a vacuum.

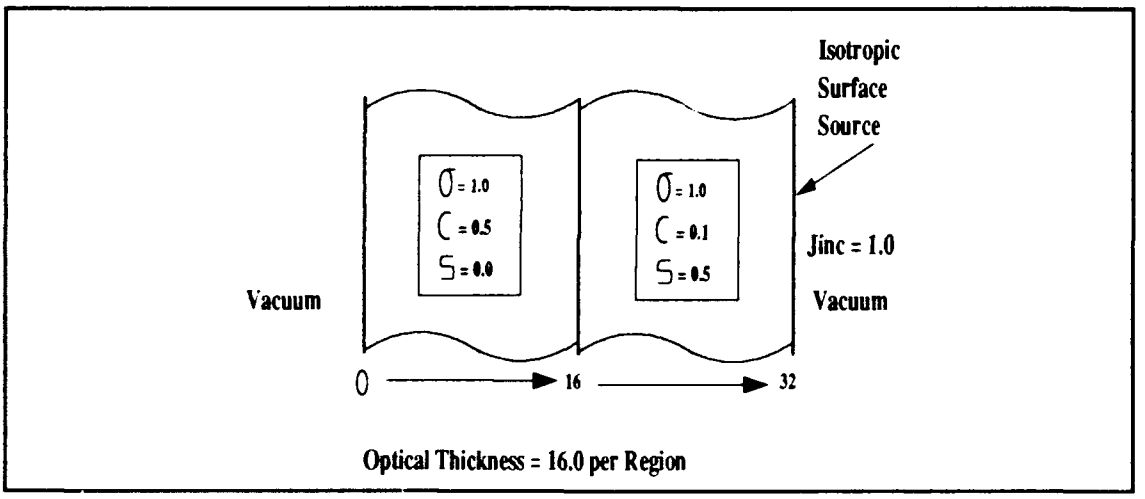

Figure 10. Schematic for Test Case 2

With moderate scattering in the left region, a modest source in the right region, and an isotropic surface source on the right boundary, the flux profile in the second (source) region should be almost flat. If this is the case, the constant b in the EC source function equation (32) will be very small, and the EC method can be expected to perform like the LC method, as discussed in Section IV.F.

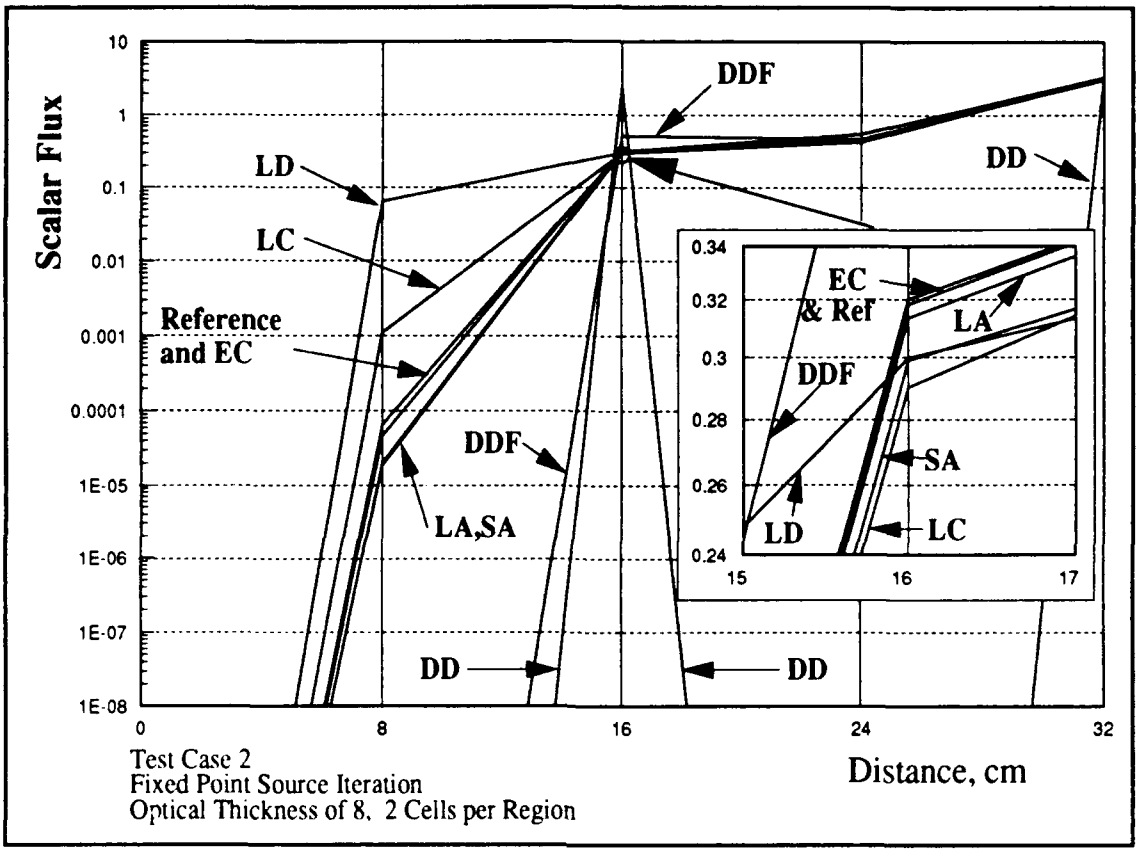

Figure 11. Test Case 2 Cell Boundary Scalar Flux

The cell boundary scalar fluxes for this problem are shown in Figure 11 using two cells per region, equivalent to an optical thickness $\sigma\Delta x = 8$ in each cell. Results of region average scalar fluxes are reported in Table 4 for this same optical thickness. As in the first problem, a reference solution was computed with linear characteristic spatial quadrature with 256 cells per region. Observe that the flux was reasonably flat in the right region between $x = 16$ and $x = 32$, as anticipated.

Moving to the left from the right edge at $x = 32$ to the cell interface in the middle of region 2 at $x = 24$, all methods appeared to yield effectively the same result except for the diamond difference method, which plunges to a meaningless negative result. Between

Table 4
Test Case 2 Region Average Scalar Fluxes for $\sigma\Delta x = 8$

Method	Region 1 ϕ_A	Region 2 ϕ_A
DD	2.8627E-02	5.7524E-01
DDF	2.8351E-02	6.0358E-01
LD	7.9753E-02	6.0516E-01
LC	1.6436E-02	6.0545E-01
SA	1.4972E-02	6.0653E-01
LA	1.5143E-02	6.0617E-01
EC	1.4876E-02	6.0631E-01
Ref	1.4646E-02	6.0550E-01

$x = 16$ and $x = 24$, DDF and especially DD were noticeably in error. At $x = 16$ where regions 1 and 2 meet, small differences in flux were reported by the various methods, as shown in the exploded portion of the graph (note how closely EC tracked along the reference solution). In the left region between $x = 0$ and $x = 16$, a wide variety of solutions resulted, indicating severe numerical diffusion from the use of optically thick cells. Remarkably, the flux solution for the exponential characteristic scheme agreed almost precisely with the reference solution throughout both regions. As in test case 1, LA offered the next best solution to EC, followed closely by SA.

Results of average pointwise and maximum observed errors are listed in Table 5 for an optical thickness of $\sigma\Delta x = 8$. At this optical thickness, the EC method yielded an average pointwise error Q less than 0.04, while this error for LA was in excess of 0.26. For the exponential characteristic scheme, the maximum error observed at any boundary or region value for $\sigma\Delta x = 8$ (including all fluxes, currents, and moments) for this problem occurred in the solution for the net current at the left boundary ($x = 0$), which was 0.14.

The maximum error for the LA scheme was found in the solution for the flux on the left boundary, which was 1.36. This further demonstrates the ability of the EC scheme to yield a reliable solution over a very coarse mesh.

Table 5
Test Case 2 Error Results for $\sigma\Delta x = 8$

Method	Average Error Q	Maximum Observed Error	Location of Maximum Observed Error
DD	0.9429	2.0000	Left Bdy ϕ
DDF	0.7580	2.0000	Left Bdy ϕ
LD	0.6742	2.0000	Region 1 ϕ_x
LC	0.4597	1.9823	Left Bdy ϕ
SA	0.2853	1.4231	Left Bdy ϕ
LA	0.2642	1.3651	Left Bdy ϕ
EC	0.0388	0.1409	Left Bdy J_{net}

A difficulty arose in the EC method for test case 2 when optical thicknesses dropped below unity during calculations in the right source region. Analysis of intermediate iteration values revealed that the first moment of the source was very small when $\sigma\Delta x < \sim 1$. The ratio of the first source moment S_x to the zeroth moment S_A was less than 6.0×10^{-6} for some cells. The value of $\beta = -b\Delta x$ was almost zero, and the number of root solving iterations required by Newton's method (See Section IV.E.iii) became unfavorably large, even for modest comparison tolerances. Although in these cases the exponential characteristic scheme is algebraically equivalent to the linear characteristic method, root solving is still required to obtain β and therefore b. When these complications arose, the execution time for the EC scheme became extremely large, burdened by the root solving overhead progressing into thousands of iterations in some

cells, as opposed to the typical two to five normally required following the initial guess. When some very optically thin cells were specified, the flux profile changed so little across some cells that no roots were found; the Newton iteration scheme oscillated the root around zero.

Since the EC method only encounters difficulty in root solving when S_x is nearly zero (compared to S_A), and since it is asymptotically equivalent to the linear characteristic method in these circumstances (See Section IV.F), a switch to LC was implemented into the EC algorithm when the initial guess for β fell to less than or equal to 1.2×10^{-5}. This value was found to be a lower limit to maximize the use of the EC method and still avoid root solving problems. The upper limit to this trip point is defined to be when the LC scheme can be actuated with no fixup. This limit is satisfied when β is as large as two. However, setting the trip point to use LC when $\beta \leq 2$ sacrifices the advantage of the EC method in many cells, and could result in oscillating or failed convergence. This is because the exponential source moments in EC can be quite unlike the linear moments in LC for values of β not close to zero, potentially resulting in iterations that flip-flop between EC and LC and thus never converge. However, this difficulty was not observed in test case 2 when the trip point for β was set to 2.

In an effort to determine an optimum set point, limited tests using test case 2 were conducted with several optical thicknesses, computed over a range of increasing initial guess setpoints for β to change to LC (between 1.2×10^{-5} and 2.0). As the set point was increased, resulting solutions became less accurate. However, between $\beta \leq 1.2 \times 10^{-5}$ and up to $\beta \leq 5.0 \times 10^{-4}$, no changes in any solutions were detected (using the standard relative tolerance of 10^{-5}), but first moment balances (using equation (26)) were conserved to a much higher level using $\beta \leq 5.0 \times 10^{-4}$. With a trip point $\beta \leq 1.2 \times 10^{-5}$, the

first moment relative tolerance was $\approx 10^{-4}$, while using $\beta \leq 5.0 \times 10^{-4}$ yielded a tolerance of $\approx 10^{-10}$. An optimum set point of $\beta \leq 5.0 \times 10^{-4}$ corresponds to an S_x/S_A of 2.5×10^{-4}. Further, this prevents extraordinarily high numbers of iterations for root solving in EC, maximizes moment balance tolerances, and actuates a switch to LC for a few cases when root solving becomes uniquely difficult (for very flat flux profiles across a cell, when $S_x \rightarrow 0$). Under these constraints, LC never requires a fixup and conserves all moments. Also, use of the LC method for an initial guess for $\beta \leq 5.0 \times 10^{-4}$ is reasonable, as the exponential characteristic scheme is asymptotically equivalent to linear characteristic in this case. As a result, solutions for test case 2 were computed using a setpoint of $\beta \leq 5.0 \times 10^{-4}$ switch to LC.

A plot of the average pointwise error using a fixed point source iteration for this problem is given in Figure 12. As shown, errors for the EC method were far lower than for any other scheme, especially for a coarse spatial mesh.

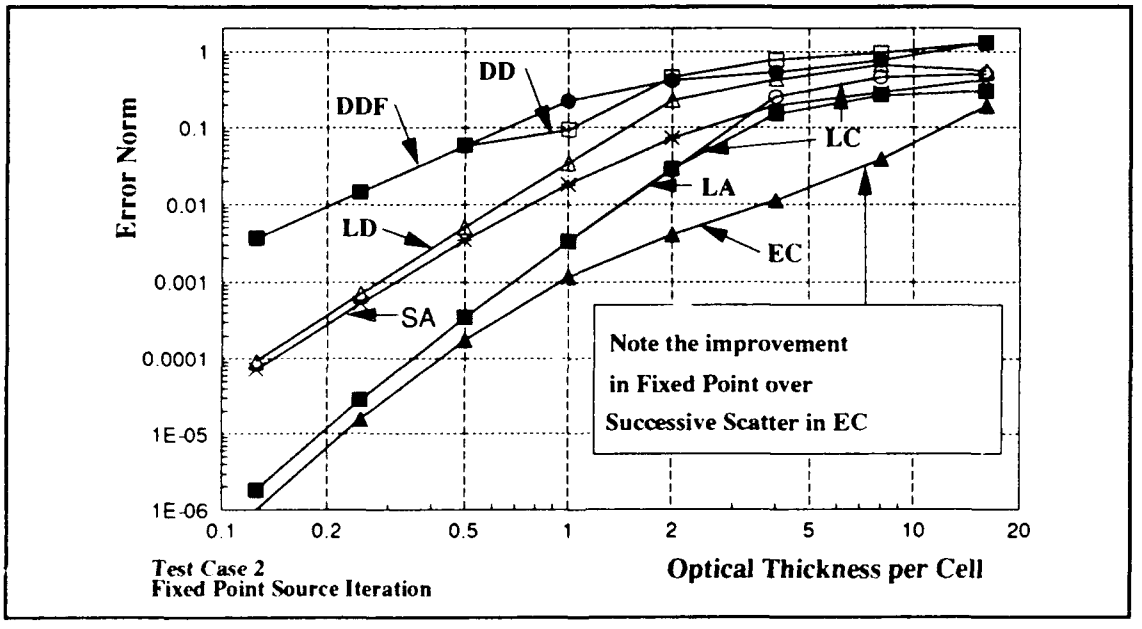

Figure 12. Test Case 2 Average Pointwise Error, Fixed Point Source Iteration

Analysis of successive scatter solutions to test case 2 demonstrated that the switch to the LC method using $\beta \leq 5.0 \times 10^{-4}$ was not necessary until optical thicknesses fell below $\sigma\Delta x = 2.0$, compared to $\sigma\Delta x = 4.0$ when fixed point source iteration was used. This result is observed because in successive scatters, the external source is only present in the source function during the initial flight of neutrons in the source region. After the first flight, the magnitude of the source function comes from only scattering of the first flight. The source term is not dominated by the constant external source (as in the fixed point case) in successive flights, resulting in ratios of first to zeroth source moments that are not as close to zero. When this occurs, β is not exceedingly small. The need to use LC only arises when the optical thickness is refined to a degree that the scattered flux for a given flight of neutrons is very flat across a cell, yielding values of S_x/S_A and β that are small.

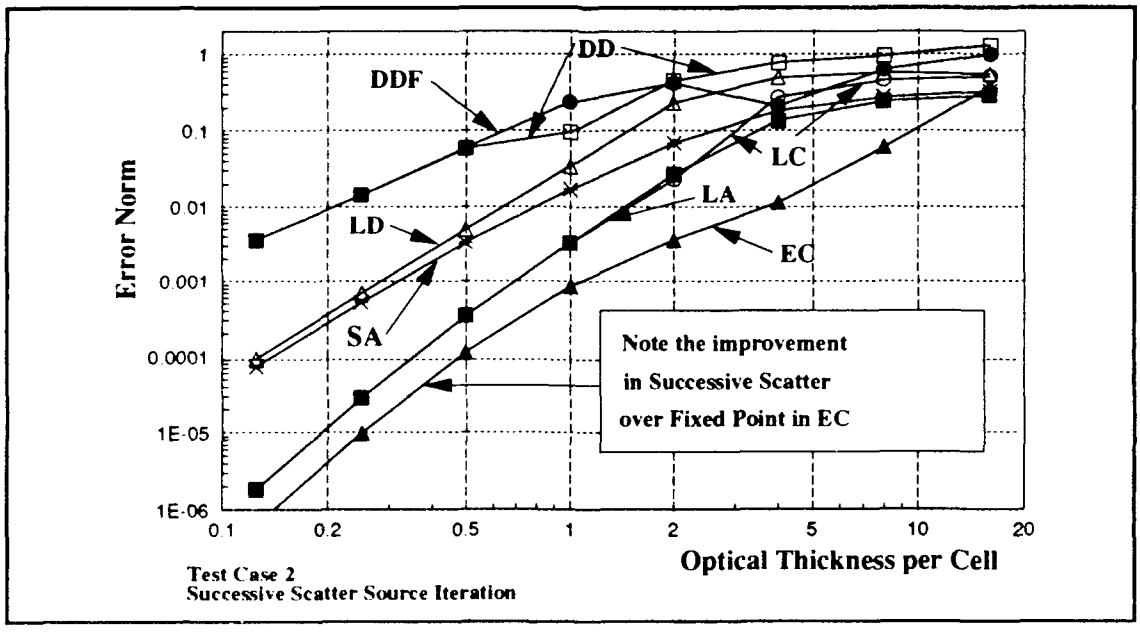

Figure 13. Test Case 2 Average Pointwise Error, Successive Scatters Source Iteration

The average pointwise errors Q for test case 2 using successive scatters are shown in Figure 13. The greatest contribution to the average error occurred for both fixed point and successive scatters at $x = 0$. Using very coarse meshes, the error using successive scatters was almost twice that of fixed point source iteration. This was not observed in the first problem which had a strongly absorbing region. Apparently, when very coarse meshes are specified with more than minimal scattering, successive scatters performs worse than fixed point iteration of the source function. At an optical thickness of two, the errors by either method are nearly equal. Below $\sigma\Delta x = 2.0$, a benefit of using successive scatters was realized with decreasing optical thickness because the source function was not dominated by the external source term; using optically thin cells, average pointwise errors for successive scatters were only ≈60 percent of those for fixed point iteration. Nevertheless, since the EC scheme delivers accurate solutions using a only a few cells, these results suggest that for this problem, the additional five iterations required for a successive scatters solution are not worth the computational effort.

Analogous to the convergence in test case 1, as the optical thickness $\sigma\Delta x \rightarrow 0$, the EC method went to 4th order in test case 2. The convergence of the LC and LA schemes trailed that of EC over coarser cells, each close to one at first, but also ending at 4th order as $\sigma\Delta x \rightarrow 0$. As expected, the diamond schemes began with very low convergence and closed on second order as the optical thickness decreased. The convergence of LD and SA showed an improvement over DD and DDF. Overall, the EC method demonstrated the highest order of convergence in the limit of thin cells.

The mesh size ratio (MSR) for test case 2 is presented in Figure 14. These results are similar to those presented in Figure 9 for test case 1. When EC used 2 cells, an MSR of nearly 4 for LC and LA resulted, indicating an equivalent of 8 cells for LC and LA.

When 4 cells were used with the exponential characteristic scheme, the linear adaptive and linear characteristic methods required 3 times as many (12 cells) to obtain the same error Q.

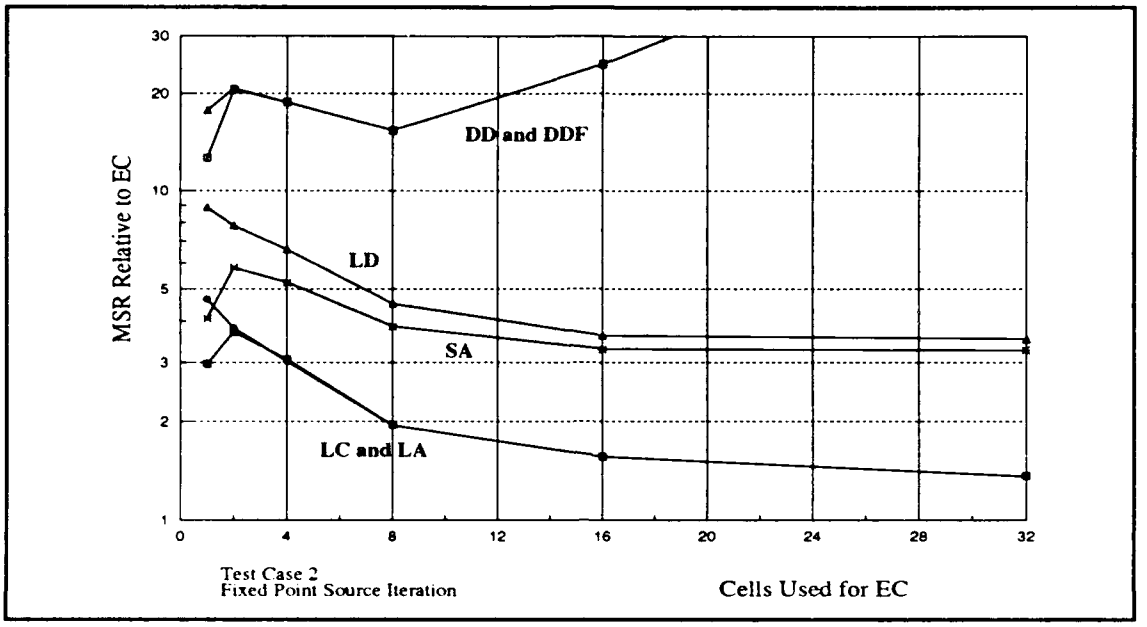

Figure 14. Test Case 2 Mesh Size Ratio

Execution time ratios (ETRs), computed using equation (39), are presented in Figure 15. Here the ETR for each spatial quadrature scheme is plotted versus average pointwise error Q, and ETR values are execution times normalized to the time used by the EC scheme. At an error Q slightly less than 4 percent, the EC scheme used 2 mesh cells, as indicated, yielding a corresponding ETR of ≈2.5 for LC and LA. Therefore, LC and LA required about 2.5 times the execution time as that consumed by the EC scheme to obtain less than a 4 percent error Q. Only when the average pointwise error fell below 0.6 percent did the execution time for EC equal that for LC or LA (using more than 15 cells

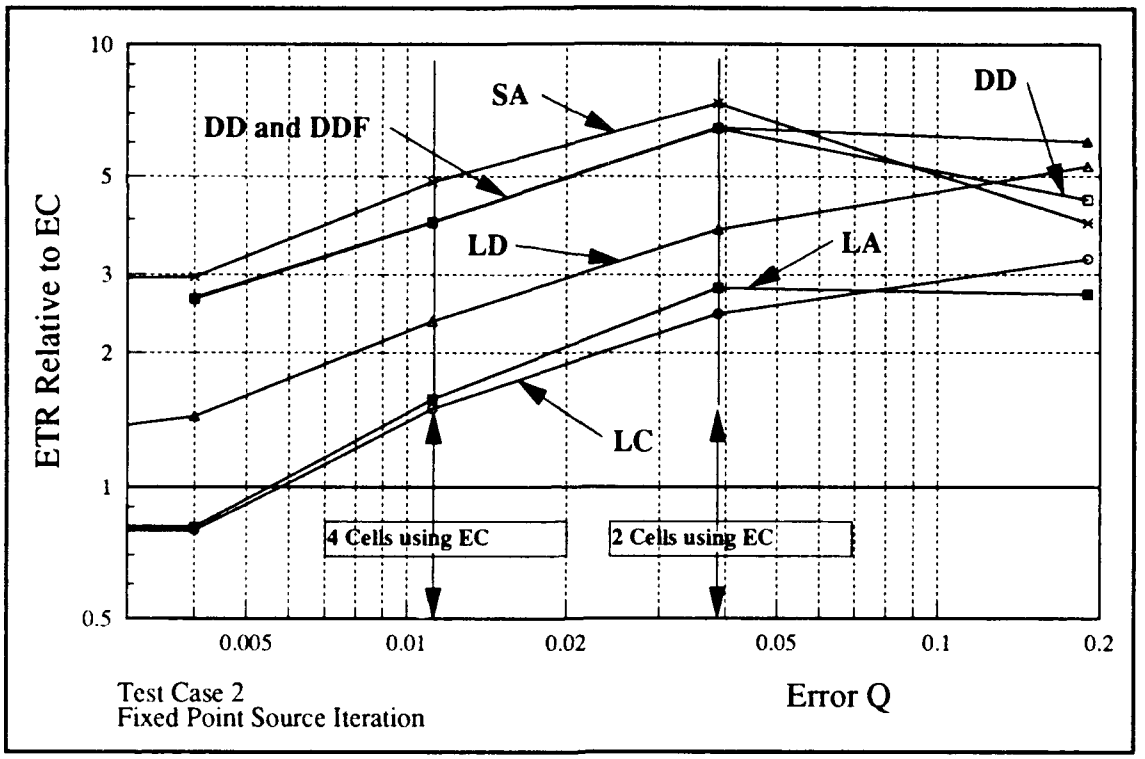

Figure 15. Test Case 2 Execution Time Ratio

each) when ≈7 cells were specified for EC. Thus, if memory storage is not a concern, either LC, LA, or EC would provide an average pointwise error of less than 0.6 percent in close to the same execution time.

C. Test Case 3: Absorber and Diffusion

Clearly, the exponential characteristic spatial quadrature scheme has demonstrated superior performance over all of the conventional schemes tested. In test case 2, the left region was a modest scatterer, and the EC scheme yielded the best overall solution, even better than LA or LC using optically thin cells. It is known that the diamond difference, linear discontinuous, and linear characteristic methods satisfy the diffusion limit; as a

diffusive problem is refined, these transport methods produce solutions that asymptotically approach the solution produced by the neutron diffusion equation (Larsen, 1982: 90-95). In diffusive problems, S_x is typically much smaller than S_A, and in the limit as the mesh is refined, the EC scheme is asymptotically equivalent to the LC scheme as $S_x \rightarrow 0$. Moreover, since the EC scheme performed better than the DD, LD, and LC schemes in the moderately diffusive region of test case 2, I anticipated that the exponential characteristic scheme would satisfy the diffusion limit. To test this expectation, test case 3 contained two regions; a strongly absorbing region on the left, and a highly diffusive region with a source on the right. Also, an isotropic surface source was located at the right boundary, and vacuum boundaries were present on either side. A schematic of this problem is given in Figure 16.

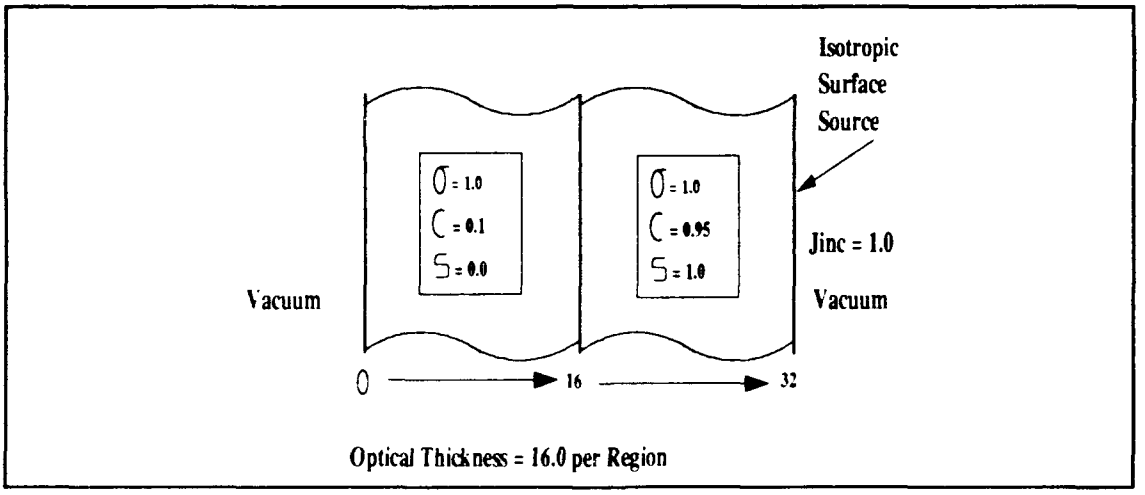

Figure 16. Schematic for Test Case 3

The solutions to the scalar flux for this problem are presented below in Figure 17. Only two cells were used to obtain these solutions; the optical thickness was $\sigma\Delta x = 8$. As expected, the flux solutions for the various methods were very close as a result of the strong diffusion of neutrons from the right region. As indicated in the figure, the

exponential characteristic scheme was not as accurate in predicting fluxes as in other problems. The exponential source function in EC has more difficulty in modeling the scalar flux in diffusive regions than the linear source function in LC. This is expected, as an exponential approximation of a concave downward flux profile is frequently more in error than for a linear approximation. Still, results for EC were found to agree closely with LC and LA.

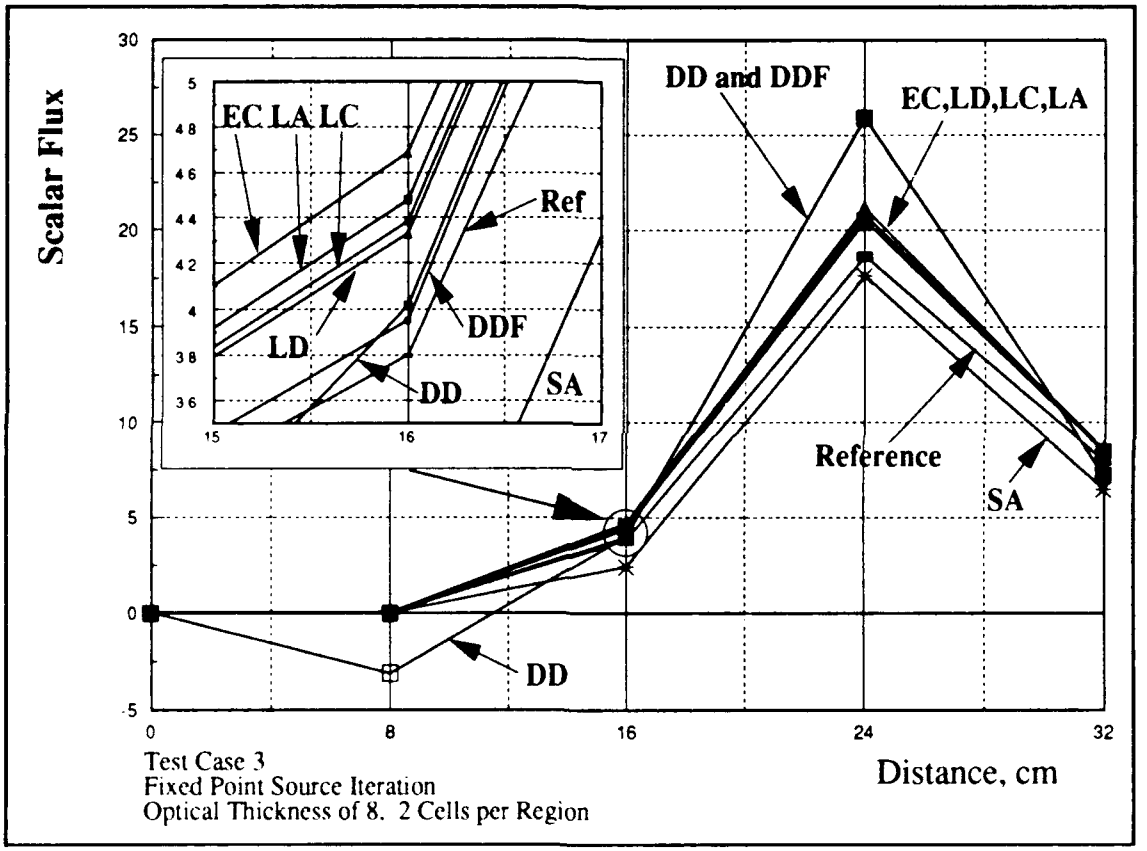

Figure 17. Test Case 3 Cell Boundary Scalar Flux

Curiously, when boundary currents, moments, and scalar fluxes are all considered, the EC scheme yielded the lowest average pointwise error Q. A plot of this average error is given in Figure 18. Note in the figure that the error Q for the LA scheme was slightly greater than that for EC.

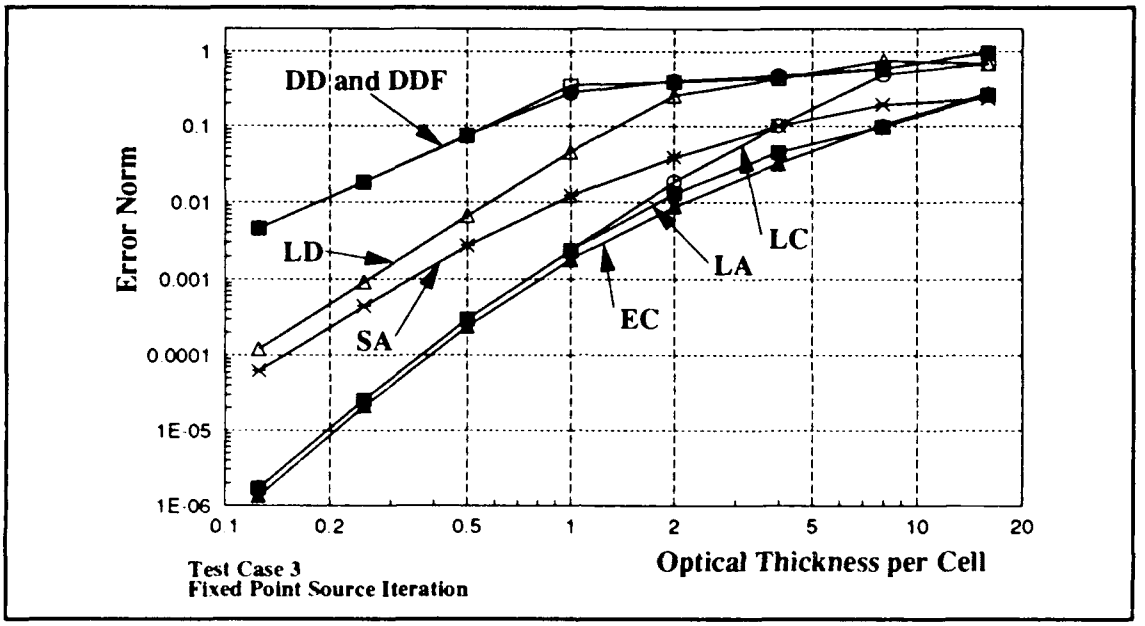

Figure 18. Test Case 3 Average Pointwise Error

Furthermore, considering the maximum observed error in any region or boundary flux, current, or moment, the EC scheme yielded very favorable results. A plot of the maximum observed error for problem 3 is presented in Figure 19. Although the maximum observed error for LA and EC are nearly the same, the average pointwise error of the exponential characteristic scheme was still lower than that for the linear adaptive method. Use of a successive scatters iteration of the source function yielded solutions that were slightly degraded for all methods. In most cases, the difference between the fixed point solution and the successive scatters solution was minor.

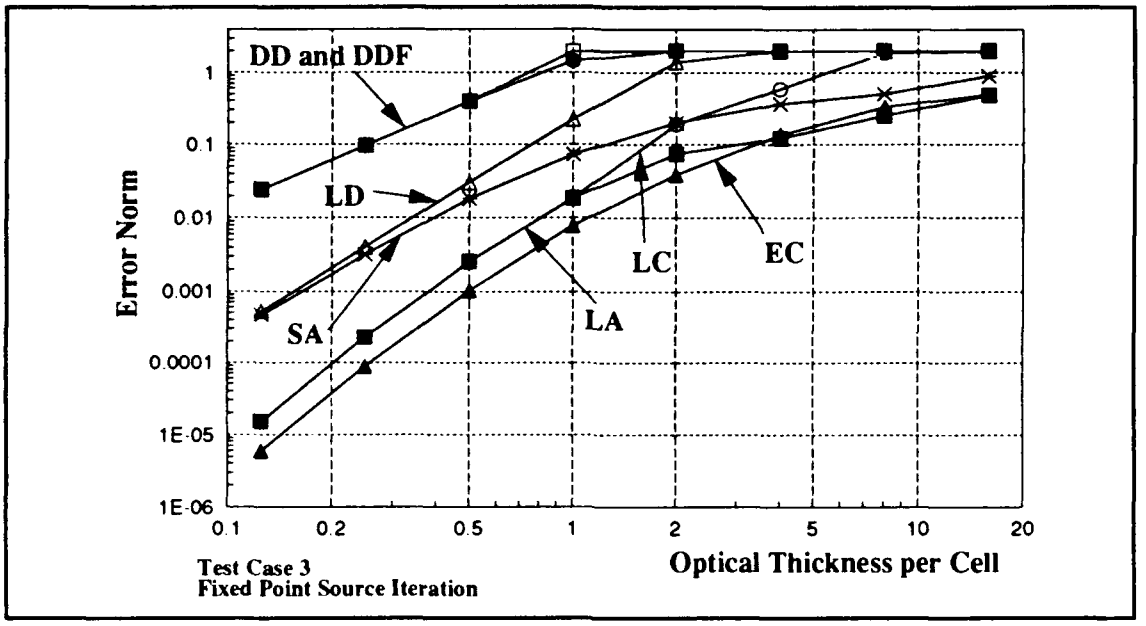

Figure 19. Test Case 3 Maximum Observed Error

The EC solution to this problem proved to be in reasonable agreement with the solutions for DD, LD, and LC methods. Of these methods, EC yielded the lowest average and maximum errors (when considering all computed flux and boundary values). Experience using the exponential characteristic method in other extremely diffusive problems showed that EC performed better globally than either DD, DDF, or LD over any optical thickness. In many cases, the solution afforded by EC was only slightly worse than that for LC or LA; again, this is likely due to the source function in LC and LA using a linear approximation instead of an exponential as in EC. Sometimes, EC yielded the best overall solution by a small margin. Although supporting evidence is extremely limited, these findings suggest that the EC method asymptotically approaches the thin cell diffusion limit.

Analysis of the mesh size ratio for test case 3 revealed that little advantage was gained in using the EC scheme over the linear characteristic or linear adaptive methods, as might be expected after inspection of the average pointwise error in Figure 18. As mentioned, solutions for the EC, LA, and LC schemes were very close. When execution time ratios were compared, the EC method was at a disadvantage, requiring more than twice the execution time of LA for the same error. Thus, on the basis of accuracy and execution time in test case 3, EC was not the most efficient scheme.

VII. Conclusions and Recommendations

From the results for test case 1, the thick absorber, the exponential characteristic scheme was superior to any other method. This is credited to the natural ability of the exponential source function to model nearly exponential attenuation. Other spatial quadratures required at least 2 to 3 times the number of cells that EC needed to yield an equivalent solution. This reveals the potential savings in memory storage offered by EC when applied to deep penetration absorbers. In two dimensions, the memory required by EC may be as little as one tenth of that necessary for most other methods for a given accuracy.

The strength of the exponential characteristic scheme was further demonstrated in test case 2. This problem included a moderate scatterer region and a source (absorbing) region, each of significant optical thickness. While most methods performed equally well in the source region with an incident current, fluxes and currents became less accurate with increasing penetration depth. Of all of the methods, the EC scheme yielded the most accurate solution over the coarsest meshes (to within a few percent of the reference solution). In addition to using fewer than 1/3 as many cells as the linear adaptive method (the closest competitor), the EC scheme required only 1/3 of the execution time required for LA to obtain an equivalent solution using a coarse mesh. Again, these slab geometry results suggest that nearly a tenfold savings in memory storage and computational cost is possible if the EC scheme is implemented in two dimensional Cartesian geometry.

The EC method encountered problems for very flat flux profiles when $S_x \rightarrow 0$, (the exponential constant β is close to zero), although this was easily solved by using the LC method in such instances. Under these circumstances, the EC method is asymptotically equivalent to the LC method.

In the strongly diffusive region of test case 3, EC solutions compared well with solutions for LC and LA. Overall, the solutions for EC were more accurate. These limited results suggest that the EC method behaves correctly in the thin cell diffusion limit. Further, although yielding a solution close to that provided by LC and LA, the EC method typically required more than twice the computational effort (mainly due to root solving requirements). Improved efficiency in root solving would decrease the disadvantage EC demonstrates here, while increasing the advantage of using EC in the other test cases.

The advantages of using successive scatters instead of a fixed point source iteration (with the exponential characteristic scheme) were not significant. The greatest benefit occurred as optical thicknesses became very thin in absorbing regions. In general, iteration of the source using successive scatters performed worse than fixed point over coarse meshes and better than fixed point over fine meshes. It is likely that the attenuation of successive flights over very coarse meshes leads to increased truncation error. Since accurate calculations are desired over very coarse cells, use of successive scatters is not advantageous.

Overall, the exponential characteristic method proved to be a robust scheme, performing best in deep penetration problems, when $|S_x| \gg S_A$. Further, considering all spatial quadrature methods used in this research, the exponential characteristic scheme typically yielded the most accurate solutions over the coarsest meshes in most problems. This was especially true in problems having several regions with various scattering and absorption properties. In most cases, the exponential characteristic method approached fourth order convergence as the mesh was refined. Thus, numerical diffusion using the EC scheme was minor. Even though the EC scheme is non-linear, no difficulties in obtaining convergence were encountered for any problem tested.

There are several issues concerning exponential characteristic spatial quadrature that warrant more attention. By far, further optimization of the root solving task offers the single largest improvement in performance for the EC method, potentially reducing total execution time required. Also, better treatment of root solving difficulties when β values are close to zero is warranted, possibly by using series expansions in β in such cases. Moreover, further investigation of the performance of the EC scheme in the diffusion limit is warranted. Finally, the utility and feasibility of conserving second spatial moments for this scheme might be investigated, using combinations of exponentials (hyperbolic forms) for the source function. In any event, this research has demonstrated that the exponential characteristic scheme is a powerful technique for solving discrete ordinates problems in slab geometry. Ultimately, the exponential characteristic scheme may be used to evaluate transport problems once thought too large or too difficult to be adequately solved on conventional computer systems.

Appendix A: Angular Quadrature

All slab geometry discrete ordinates transport problems were solved using S_8 Gauss-Legendre angular quadrature. Quadrature direction cosines and weights are given in Table 6, with a graphical representation in Figure 20.

Table 6
Eight Point Single Range Gauss-Legendre Quadrature Constants

m	$\approx \theta_m^\circ$	$\approx \theta_{mRad}$	$\mu_m = \mathrm{Cos}(\theta_m)$	w_m
1	163.8	0.905 π	-.960289856497536	.101228536290376
2	142.8	0.793 π	-.796666477413627	.222381034453374
3	121.7	0.676 π	-.525532409916329	.313706645877877
4	100.6	0.559 π	-.18343464249565	.362683783378362
5	79.4	0.441 π	+.18343464249565	.362683783378362
6	58.3	0.324 π	+.525532409916329	.313706645877877
7	37.2	0.207 π	+.796666477413627	.222381034453374
8	16.2	0.090 π	+.960289856497536	.101228536290376

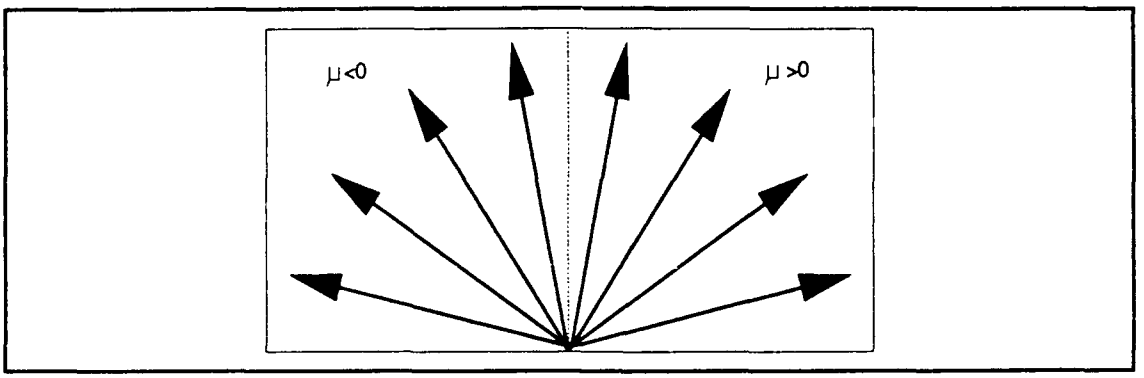

Figure 20. Direction Cosines for S_8 Angular Quadrature

Appendix B: Boundaries and Incident Currents

Descriptions of three types of boundaries and three kinds of incident currents available in the multi-method fixed point and successive scatters discrete ordinates codes are provided here. More detailed descriptions of each are available in the literature (Lewis and Miller, 1984). Boundary conditions made available in the discrete ordinates codes SN1D-ALL and SS1D-ALL are a vacuum, a symmetric albedo, and a grey albedo. Incident left or right boundary currents available are a Lambertian current, an isotropic surface source current, or a collimated beam current.

i. Vacuum Boundary

If a vacuum boundary is specified, this means that the area beyond the last physical region defined for the problem is a vacuum. No neutrons can be scattered or reflected back into the problem after crossing the physical boundary. This is often called a "zero return current" condition.

ii. Symmetric Albedo Boundary

At a symmetric albedo boundary, for each amount of flux leaving across this boundary, an albedo α (a fraction between 0 and 1) of the original outbound flux enters in a direction corresponding to a spectral reflection. A symmetry boundary is a special case of the symmetric albedo boundary, where the albedo $\alpha = 1.0$; this is often specified when a completely symmetrical problem is specified and a solution for only half of the entire problem is necessary.

iii. Grey Albedo Boundary

The grey albedo boundary holds that an albedo α times the outbound flux of neutrons passing across the surface of the boundary return back into the boundary in an isotropic distribution, uniformly dispersed over all angles. This is accomplished in slab geometry by noting, for instance, at a left boundary, the entering (positive) current is given by

$$J^{+}(0)=\frac{1}{\alpha}\int_{0}^{1}\mu\psi(0,\mu)\frac{d\mu}{2} \tag{84}$$

If the incoming flux is a constant value $\psi^{+}(0)=A$ (as specified by the isotropic condition) for directions $0\le\mu\le 1$, then the positive current reduces to

$$J^{+}(0)=\frac{A}{4\alpha} \tag{85}$$

From continuity of current at the boundary, $J^{+}(0)=J^{-}(0)$, which yields

$$\psi^{+}(0)=A=4\alpha J^{-}(0) \tag{86}$$

This states that for a grey albedo left boundary, the outgoing (negative) current is multiplied by four times the albedo factor α to yield an isotropic incoming angular flux over $0\le\mu\le 1$.

A special case of the grey albedo boundary is a white boundary, where $\alpha = 1$, and all exiting current returns in an isotropic distribution.

iv. Lambertian Current

A Lambertian incident current is due to particle emissions from an outside isotropic source, so that the angular flux which contributes to a Lambertian current is a constant F. Using a left boundary and integrating over positive angles, the following equation yields the incident current :

$$J^{+}_{incL}(0)=\frac{F}{4} \tag{87}$$

Thus, the incident current attributable to each direction cosine is

$$J_{inc}(\mu) = \mu 4 J^{+}_{incL} \tag{88}$$

A similar treatment is made at the right boundary.

v. Isotropic Surface Source Current

An isotropic surface source is a planar source of isotropic emitters, where the product of the angular flux and direction cosine is held constant. Therefore, when the direction cosine μ is close to zero, the angular flux is highest due to greatest contribution from the isotropic surface sources. Subsequently, the incident current attributable to each direction cosine at a left boundary is

$$J_{inc}(\mu) = 2 J^{+}_{incL} \tag{89}$$

vi. Collimated Beam Current

Discrete ordinates methods often do not accurately deal with highly anisotropic flux distributions, e.g. a collimated beam. This is because the ordinates of a standard

quadrature set, symmetric from -1 to 1, do not individually provide an accurate account of the forward angular flux from an incident current. Under these circumstances, the best method of treating a collimated beam incident in a direction cosine μ' that is not part of the quadrature set is to interpolate the incident direction between applicable directions in the quadrature set. Thus, the incident current in μ' is shared between μ_i and μ_{i+1}, where quadrature weights and direction cosines are used to adjust the magnitude of the incident current to yield accurate quadratures.

Appendix C: Test Case Problems

Presented below are the test cases and their reference solutions using fixed point source iteration (solutions obtained using successive scatters are similar). These reference solutions were used to illustrate the performance of the exponential characteristic method in Section VI. Only test case 1 was acquired using an 8 MHz 8086 personal computer with 8087 math support. All other problems were solved using a 33 MHz 80486 ISA machine. (The 80486 machine tested approximately twenty times faster than the 8086 machine).

i. Test Case 1: Thick Absorber

The input parameters and reference solution for the single region thick absorber problem are given below:

Problem file: A:TMA1.IPT Output file: B:TMA1-256.OUT

```
Ident$: MdlP Mathews 90
number of regions = 1
left bdy position = 0
type of left bdy = 0  vacuum
current incident at left boundary = 1
type of current incident at left bdy = 0 isotropic surface Src
region#   cR      SigmaR      SourceR     nc   Right Bdy
  1     0.1000  1.0000D+00  0.0000D+00   256.  16.0000
type of right bdy = 0  vacuum
current incident at right boundary = 0
type of current incident at right bdy =-1 Lambertian
```

```
Problem file   : A:TMA1.IPT      nk Angular Ordinates: 8
Output file    : B:TMA1-256.OUT  Quadrature Method   : S8
Tk file        : B:TMA1-256.TKD  Negative Flux Fixups: NO
Max Iterations : 150             Reset Old Fluxes    : YES
File Identifier: MdlP Mathews 90 Solution Tolerance  : 1.00D-05
```

```
00:45:20 11-01-1991 SN1D-ALL.BAS Version 2.00 - 30 Oct 1991

Linear Characteristic -> LC Source Rotation (Sx<=Sa) Fixup ALWAYS Enabled
Execution Time :  2.4488 min
Converged After 8 iterations, MaxChangeObs = 9.686617141449785D-06

     J plus       J minus      J net        Bdy Flux
Reg#  flux ave     flux x-moment

    +1.000000D+00 +2.633397D-02 +9.736660D-01 +3.022184D+00
 1  +6.761570D-02 -1.892749D-01
    +8.612159D-09 +0.000000D+00 +8.612159D-09 +9.517948D-09

Mom Bals: 0th: CONSERVED  MaxRelErr: 4.4702D-13≤ 1.0000D-10
      1st: VIOLATION          1.4321D-10≤ 1.0000D-10
 Region Bals: CONSERVED          3.6304D-08≤ 1.0000D-05
```

ii. Test Case 2: Moderate Scatter and Source

The input parameters and reference solution for the moderate scatter and absorber-source problem are given below:

```
Problem file: D:TGS1.IPT  Output file: D:TGS1-256.OUT

Ident$: Ges 16mfp Abs+Src
number of regions = 2
left bdy position = 0
type of left bdy = 0  vacuum
current incident at left boundary = 0
type of current incident at left bdy = 0 isotropic surface Src
region#   cR      SigmaR      SourceR   nc  Right Bdy
  1     0.5000  1.0000D+00  0.0000D+00  256.  16.0000
  2     0.1000  1.0000D+00  5.0000D-01  256.  32.0000
type of right bdy = 0  vacuum
current incident at right boundary = 1
type of current incident at right bdy = 0 isotropic surface Src
```

```
Problem file   : D:TGS1.IPT      nk Angular Ordinates: 8
Output file    : D:TGS1-256.OUT  Quadrature Method   : S8
Tk file        : D:TGS1-256.TKD  Negative Flux Fixups: NO
Max Iterations : 150             Reset Old Fluxes    : YES
File Identifier: Ges16mfp Abs+Src Solution Tolerance : 1.00D-05
```

```
20:17:13 11-23-1991 SN1D-ALL.BAS Version 2.00 - 30 Oct 1991

Linear Characteristic -> LC Source Rotation (Sx<=Sa) Fixup ALWAYS Enabled
Execution Time :  1.0911 min
Converged After 24 iterations, MaxChangeObs = 9.771934993029279D-06

     J plus       J minus      J net        Bdy Flux
Reg#  flux ave     flux x-moment

    +0.000000D+00 +1.693132D-08 -1.693132D-08 +2.303910D-08
 1   +1.464628D-02 +3.908373D-02
    +2.065823D-02 +1.378285D-01 -1.171702D-01 +3.183050D-01
 2   +6.054969D-01 +1.854079D-01
    +1.636746D-01 +1.000000D+00 -8.363254D-01 +3.292646D+00

Mom Bals: 0th: CONSERVED   MaxRelErr: 1.5519D-13≤ 1.0000D-10
      1st: VIOLATION           1.0979D-07≤ 1.0000D-10
 Region Bals: CONSERVED          2.0834D-08≤ 1.0000D-05
```

iii. Test Case 3: Absorber and Diffusion

The input parameters and reference solution for the absorber and diffusion problem are given below:

```
Problem file: D:TGS2.IPT   Output file: D:TGS2-256.OUT
```

```
Ident$: Ges 16mfp Abs+Diff
number of regions = 2
left bdy position = 0
type of left bdy = 0  vacuum
current incident at left boundary = 0
type of current incident at left bdy = 0   isotropic surface Src
reg #   cR      SigmaR      SourceR   nc   Right Bdy
  1   0.1000  1.0000D+00  0.0000D+00  256.  16.0000
  2   0.9500  1.0000D+00  1.0000D+00  256.  32.0000
type of right bdy = 0  vacuum
current incident at right boundary = 1
type of current incident at right bdy = 0 isotropic surface Src
```

```
Problem file   : D:TGS2.IPT      nk Angular Ordinates: 8
Output file    : D:TGS2-256.OUT  Quadrature Method   : S8
Tk file        : D:TGS2-256.TKD  Negative Flux Fixups: NO
Max Iterations : 200             Reset Old Fluxes    : YES
File Identifier: Ges16mfp Abs+Dif Solution Tolerance : 1.00D-05
```

```
03:55:40 12-02-1991 SN1D-ALL.BAS Version 2.00 - 30 Oct 1991

Linear Characteristic -> LC Source Rotation (Sx<=Sa) Fixup ALWAYS Enabled
Execution Time :  6.5388 min
Converged After 144 iterations, MaxChangeObs = 9.652863571151075D-06

     J plus       J minus      J net        Bdy Flux
Reg #  flux ave     flux x-moment

    +0.000000D+00 +3.550017D-08 -3.550017D-08 +3.913293D-08
 1   +1.386661D-01 +3.786462D-01
    +4.430641D-02 +2.041100D+00 -1.996793D+00 +3.803869D+00
 2   +1.544256D+01 +9.081632D-01
    +2.647280D+00 +1.000000D+00 +1.647280D+00 +7.970842D+00

Mom Bals: 0th: CONSERVED   MaxRelErr: 8.9809D-13≤ 1.0000D-10
      1st: VIOLATION           4.4807D-07≤ 1.0000D-10
 Region Bals: CONSERVED          7.5017D-06≤ 1.0000D-05
```

Appendix D: Source Code

The following is the source listing for the multi-method, slab geometry, fixed point source iteration SN1D-ALL code, written in MicroSoft QuickBASIC 4.5. Except for the exponential characteristic scheme implemented by the author, all methods were derived from individual codes originally written by LCDR K. Mathews, Ph.D. at the Department of Engineering Physics, Air Force Institute of Technology, Wright Patterson AFB, OH, 45324. As discussed, the author incorporated all codes into this single compilation for a uniform analysis approach. The successive scatter transport code, SS1D-ALL, and the error compilation code, PROCSNO2, are archived with LCDR Mathews at the above address.

```
DECLARE SUB SetSwitchECtoLC ()
DECLARE SUB StepDD (Jin#,Jout#,Sx#,dx#,Sigma#,mu#.Fa#)
DECLARE SUB StepSC (Sigma#,DeltaX#,mu#,Fl#,Sa#,Sx#,Fr#,Fa#,Fx#)
DECLARE SUB StepLC (Sigma#,DeltaX#,mu#,Fl#,Sa#,Sx#,Fr#,Fa#,Fx#)
DECLARE SUB StepLN (Sigma#,DeltaX#,mu#,Fl#,Sa#,Sx#,Fr#,Fa#,Fx#)
DECLARE SUB StepSA (Sigma#,DeltaX#,mu#,Fl#,Sa#,Sx#,Fr#,Fa#,Fx#)
DECLARE SUB StepLA (Sigma#,DeltaX#,mu#,Fl#,Sa#,Sx#,Fr#,Fa#,Fx#)
DECLARE SUB StepEC (Sigma#,DeltaX#,mu#,Fl#,Sa#,Sx#,Fr#,Fa#,Fx#)
DECLARE SUB Mom0StepCheck (Fl#,Fr#,Fa#,eps#,Sa#,DeltaX#,mu#)
DECLARE SUB Mom1StepCheck (Fl#,Fr#,Fa#,Fx#,eps#,Sx#,DeltaX#,mu#)
DECLARE SUB PrtTK (labl$, vrbl#(), istart%, n%)
DECLARE SUB BeginSets ()
DECLARE SUB AltMenu ()
DECLARE SUB makeP (p#(), e#)
DECLARE FUNCTION MIN# (x#, y#)
DECLARE FUNCTION MAX# (x#, y#)
DECLARE FUNCTION Choose% (prompt$)
DECLARE FUNCTION ParseNthWord$ (a$, n%)
DECLARE FUNCTION Gx# (x#, ro#)
DECLARE FUNCTION DdxG# (x#)

CONST False = 0
CONST True = NOT False

DEFINT I, K, N
DEFDBL A-G, J, L-M, O-Z

pi = 4 * ATN(1!)

'PROGRAM SN1D-ALL.BAS
'Discrete Ordinates (SN) 1D Slab Geometry Program
'for research at the Air Force Institute of Technology
'by Kirk A. Mathews, Ph.D, modified by Glenn E. Sjoden, P.E.

V$ = "SN1D-ALL.BAS Version 2.10 - 10 Dec 1991"
```

```
'-- Spatial Quadrature Methods Supported in this Code --
'     DD-Diamond Difference       (from V-2.05, 19 Feb 90)
'     LD-Linear Discontinuous     (from V-2.04,  4 Feb 90)
'     SC-Step Characteristic      (from V-2.04,  1 Mar 90)
'     LC-Linear Characteristic    (Modified from LA below)
'     LN-Linear Nodal             (from V-2.04,  4 Feb 90)
'     SA-Step Adaptive            (from V-2.04,  4 Feb 90)
'     LA-Linear Adaptive          (from V-2.04,  4 Feb 90)
'       - by Kirk A. Mathews, Ph.D.

'     EC-Exponential Characteristic   (V-1.20, 16 Oct 91)
'       - by Glenn E. Sjoden, P.E.

WIDTH LPRINT 80
'Arrays used in defining the problem --
'  Xbdy(ir) is right bdy of region ir
'  cR(ir) is scatter to total cross-section ratio in region ir
'  SigmaR(ir) is total cross-section in region ir
'  source(ir) is volumetric source in region ir
'  nc(ir) is number of cells in region ir
'  mu(k) is k'th "discrete ordinate" (direction in quadrature set)
'  w(k) is weight associated with mu(k) in angular quadrature set
'  dx(ix) is thickness of cell ix
'  Fa(ix,k) is flux (psi) in direction mu(k) averaged over cell ix
'  Fx(ix,k) is first moment of psi in direction mu(k) over cell ix
'  J(ix,k) is current (mu*psi) in direction mu(k)
'        through right side of cell ix
'     Note: j is positive for positive mu and neg. for neg. mu
'  FluxA(ix) is scalar flux averaged over cell ix
'  FluxX(ix) is first moment of scalar flux over cell ix
'  sigma(ix) is total cross-section in cell ix
'  c(ix) is scatter to total cross-section ratio in cell ix
'  source(ix) is volumetric source in cell ix
'  Sa(ix) is average source term (s = c*flux+source/sigma) in cell ix
'  Sx(ix) is change in Sa across cell ix
'  lprint ix, nx
 Problemfile$ = ParseNthWord((COMMAND$), 1)
 'returns first parameter found on command line
 DO
    GOSUB CommandProfile
    SELECT CASE CmdProf$
    CASE "S"
       GOSUB OpenProblemFile
       NewProblem% = True
       SwECtoLCSetPt = .000012
       SwitchEC% = False
       GOSUB ReadProblemData
       CLOSE #1  'close input file
       DevEcho$ = "SCREEN"
       DO
          CLS
          GOSUB EchoProblem
          GOSUB AdjustProblemData
       LOOP UNTIL EnterNC% = False
       ff% = Choose("Use Form Feeds in Summaries? (Y/N): ")
       DevEcho$ = "LPT1"
       GOSUB ValidateProblem
       PRINT
       IF Choose("Echo Input Data? (Y/N): ") THEN
          DevEcho$ = "LPT1"
          GOSUB EchoProblem
       END IF
```

```
      GOSUB ResetMomerrs
      DO
         GOSUB InputMethodParameters
         GOSUB SetClocks
         DevEcho$ = "LPT1"
         GOSUB EchoMethod
         DevEcho$ = "SCREEN"
         GOSUB EchoMethod
         GOSUB InitializeProblem
         NewProblem% = False
         GOSUB SnSolve
         GOSUB CalculateResults
         GOSUB VerifyRegionsByBalanceEquation
         StopClock = TIMER
         ExecMin = (StopClock - StartClock) / 60!
         DevEcho$ = "LPT1"
         GOSUB LPrintRegionSummary
         GOSUB CreateTKDataFile
         GOSUB ResetMomerrs
         SwitchEC% = False
      LOOP WHILE Choose("Solve same problem with different Sn? (y/n):
")
      PRINT
      Problemfile$ = ""
   CASE "A"
      ff% = False
      CALL BeginSets
      CALL AltMenu
      NewProblem% = True
      GOSUB ReadProblemData
      CLOSE #1  'close input file
      DevEcho$ = "SCREEN"
      DO
         CLS
         GOSUB EchoProblem
         GOSUB AdjustProblemData
      LOOP UNTIL EnterNC% = False
      DevEcho$ = "SCREEN"
      GOSUB ValidateProblem
      IF UCASE$(Qoutf$) = "Y" THEN
         DevEcho$ = "OUT"
         GOSUB EchoProblem
      END IF
      GOSUB ResetAgain
RefluxA:
 DO
   IF again% THEN
      CALL AltMenu
      DevEcho$ = "SCREEN"
      DO
         CLS
         GOSUB EchoProblem
         GOSUB AdjustProblemData
      LOOP UNTIL EnterNC% = False
      DevEcho$ = "SCREEN"
      GOSUB ValidateProblem
      IF UCASE$(Qoutf$) = "Y" THEN
         DevEcho$ = "OUT"
         GOSUB EchoProblem
      END IF
      GOSUB ResetAgain
   END IF
```

```
      GOSUB InputMethodParameters
      GOSUB SetClocks
      DevEcho$ = "SCREEN": Eout$ = "FULL"
      GOSUB EchoMethod
      LOCATE 12, 12: PRINT "Processing -> "; TotDifMeth$; " <-"
      IF UCASE$(Qoutf$) = "Y" THEN
         DevEcho$ = "OUT"
         IF itim = 0 THEN
            Eout$ = "FULL"
         ELSE
            Eout$ = "PARTIAL"
         END IF
         GOSUB EchoMethod
      END IF
      GOSUB InitializeProblem
      IF ResetAllFluxes$ = "YES" THEN
         NewProblem% = True
      ELSE
         NewProblem% = False
      END IF
      GOSUB SnSolve
      GOSUB CalculateResults
      GOSUB VerifyRegionsByBalanceEquation
      StopClock = TIMER
      ExecMin = (StopClock - StartClock) / 60!
      GOSUB Incremitim
      DevEcho$ = "SCREEN": Eout$ = "FULL"
      GOSUB EchoMethod
      GOSUB LPrintRegionSummary
      IF UCASE$(Qoutf$) = "Y" THEN
         DevEcho$ = "OUT"
         GOSUB LPrintRegionSummary
      END IF
      GOSUB CreateTKDataFile
      GOSUB ToggleDifMeth
      GOSUB ResetMomerrs
   LOOP WHILE INSTR("DDLDSCLCLNSALAEC", DifMeth$) AND LEN(DifMeth$) = 2
   DevEcho$ = "SCREEN"
   GOSUB RunSummary
   IF UCASE$(Qoutf$) = "Y" THEN
      DevEcho$ = "OUT"
      GOSUB RunSummary
   END IF
   BEEP: BEEP
   IF Choose("Solve same problem with different Sn? (y/n): ") THEN
      again% = True
      GOTO RefluxA
   END IF
END SELECT
LOOP WHILE Choose("Solve Another Problem? (y/n): ")
END

CommandProfile:
 CLS : PRINT V$: PRINT
 DO
   INPUT "Type of Command Profile (S-Standard/ A-Alternate) ", CmdProf$
   CmdProf$ = UCASE$(LEFT$(LTRIM$(CmdProf$), 1))
 LOOP UNTIL INSTR("SA", CmdProf$) AND LEN(CmdProf$) = 1
RETURN
```

```
SetClocks:
 StartDate$ = DATE$: StartTime$ = TIME$
 StartClock = TIMER: CLS
RETURN

ResetAgain:
 again% = False
 SwitchEC% = False
 REDIM Rtim(1 TO 8), RelRtim(1 TO 8)
 REDIM RErranal(1 TO 8), RelRErranal(1 TO 8)
 itim = 0
RETURN

Incremitim:
 itim = itim + 1
 Rtim(itim) = ExecMin
 RErranal(itim) = MaxRegrerr
 CLS
RETURN

ResetMomerrs:
 Mom0IntFlag$ = " 0th: CONSERVED": MaxMom0rerr = 0
 Mom1IntFlag$ = " 1st: CONSERVED": MaxMom1rerr = 0
RETURN

ToggleDifMeth:
 ilotogl = ilotogl + 3: ihitogl = ihitogl + 3
 DifMeth$ = LTRIM$(MID$(TotDifMeth$, ilotogl, ihitogl))
 DifMeth$ = UCASE$(LEFT$(LTRIM$(DifMeth$), 2))
RETURN

OpenProblemFile:
 CLS
 PRINT V$
 PRINT
 IF Problemfile$ = "" THEN INPUT "File to read for input: ",
Problemfile$
 OPEN Problemfile$ FOR INPUT AS #1
 'check for correct file:
 INPUT #1, Ident$
 PRINT Ident$
 DO
   INPUT "Correct File? (Yes/No/Quit): ", a$
   a$ = UCASE$(LEFT$(LTRIM$(a$), 1))
 LOOP UNTIL INSTR("YNQ", a$) AND LEN(a$) = 1
 SELECT CASE a$
    CASE "N"
       CLOSE #1
       Problemfile$ = ""
       GOTO OpenProblemFile
    CASE "Q"
       END
    CASE "Y"
       CLS
    CASE ELSE
       BEEP
       PRINT "ERROR: Unsupported choice in OpenProblemFile."
       STOP
 END SELECT
RETURN
```

```
ReadProblemData:
 INPUT #1, nr        'number of regions
 REDIM Xbdy(nr), cR(nr), SigmaR(nr), SourceR(nr), nc(nr)
 INPUT #1, Xbdy(0)       'left bdy position
 INPUT #1, tlb           'type of left bdy --
 '0 = vacuum
 '(0,+1] = symmetric albedo
 '{specular reflection factor}
 '[-1,0) = grey albedo
 '{Lambertian reflection factor}
 INPUT #1, jinclb        'current incident at left boundary
 INPUT #1, tinclb        'type of current inc at left bdy --
 '-1 = Lambertian
 '0 = isotropic surface source
 '(0,+1] = abs(mu) of collimated incident beam
 FOR ir = 1 TO nr
    INPUT #1, cR(ir), SigmaR(ir), SourceR(ir), nc(ir), Xbdy(ir)
 NEXT ir
 INPUT #1, trb, jincrb, tincrb
RETURN

AdjustProblemData:
 EnterNC% = Choose("Manually enter values for # cells / region? (y/n):
")
 IF EnterNC% THEN
    FOR ir = 1 TO nr
       PRINT " "; ir; TAB(10);
       PRINT USING "##.####"; cR(ir);
       PRINT TAB(20);
       PRINT USING "##.####^^^^"; SigmaR(ir);
       PRINT TAB(34);
       PRINT USING "##.####^^^^"; SourceR(ir);
       PRINT TAB(48);
       PRINT USING "-> ###. <-"; nc(ir);
       PRINT TAB(58);
       PRINT USING "###.####"; Xbdy(ir)
       IF EnterNC% THEN
          PRINT "Region"; ir;
          INPUT "Number of cells?   nc = ", nc(ir)
          PRINT
       END IF
    NEXT ir
 END IF
 PRINT
RETURN

ValidateProblem:
 bad% = False
 SELECT CASE DevEcho$
 CASE "LPT1"
    OPEN "LPT1:" FOR OUTPUT AS #4
 CASE "SCREEN"
    OPEN "SCRN:" FOR OUTPUT AS #4
 CASE "OUT"
    OPEN Outfile$ FOR APPEND AS #4
 END SELECT
 FOR ir = 1 TO nr
    IF Xbdy(ir) <= Xbdy(ir - 1) THEN
       PRINT #4, "Bad Input:"
       PRINT #4, "Xbdy("; ir; ") = "; Xbdy(ir)
       bad% = True
    END IF
```

```
      IF cR(ir) < 0 OR cR(ir) > 1 THEN
         PRINT #4, "Bad Input:"
         PRINT #4, "cR("; ir; ") = "; cR(ir)
         bad% = True
      END IF
      IF SigmaR(ir) <= 0 THEN
         PRINT #4, "Bad Input:"
         PRINT #4, " SigmaR("; ir; ") ="; SigmaR(ir)
         bad% = True
      END IF
      IF SourceR(ir) < 0 THEN
         PRINT #4, "Bad Input:"
         PRINT #4, " SourceR("; ir; ") ="; SourceR(ir)
         bad% = True
      END IF
      IF nc(ir) <= 0 THEN
         PRINT #4, "Bad Input:"
         PRINT #4, " nc("; ir; ") ="; nc(ir)
         bad% = True
      END IF
   NEXT ir
   IF ABS(tlb) > 1 THEN
      PRINT #4, "Bad Input:"
      PRINT #4, " type of left boundary -- tlb ="; tlb
      bad% = True
   END IF
   IF ABS(trb) > 1 THEN
      PRINT #4, "Bad Input:"
      PRINT #4, " type of right boundary -- trb ="; trb
      bad% = True
   END IF
   IF jinclb < 0 THEN
      PRINT #4, "Bad Input:"
      PRINT #4, " negative incident current at left -- jinclb ="; jinclb
      bad% = True
   END IF
   IF jincrb < 0 THEN
      PRINT #4, "Bad Input:"
      PRINT #4, " negative incident current at right -- jincrb ="; jincrb
      bad% = True
   END IF
   IF tinclb < -1 OR (tinclb > -1 AND tinclb < 0) OR tinclb > 1 THEN
      PRINT #4, "Bad Input:"
      PRINT #4, " type of incident current at left -- tinclb ="; tinclb
      bad% = True
   END IF
   IF tincrb < -1 OR (tincrb > -1 AND tincrb < 0) OR tincrb > 1 THEN
      PRINT #4, "Bad Input:"
      PRINT #4, " type of incident current at right -- tincrb ="; tincrb
      bad% = True
   END IF
   IF bad% = True AND (DevEcho$ = "LPT1") THEN
      PRINT #4, CHR$(12)
      CLOSE #4
      END
   ELSEIF bad% = True THEN
      CLOSE #4
      END
   END IF
   CLOSE #4
RETURN
```

```
EchoProblem:
 SELECT CASE DevEcho$
 CASE "LPT1"
    OPEN "LPT1:" FOR OUTPUT AS #4
 CASE "SCREEN"
    OPEN "SCRN:" FOR OUTPUT AS #4
 CASE "OUT"
    OPEN Outfile$ FOR APPEND AS #4
 END SELECT
 PRINT #4, "*****************************************************************
***************"
 PRINT #4, "Problem file: "; UCASE$(Problemfile$); TAB(44); "Output
file: "; UCASE$(Outfile$)
 PRINT #4, "*****************************************************************
***************"
 PRINT #4,
 PRINT #4, "Ident$: "; Ident$
 PRINT #4, "number of regions ="; nr
 PRINT #4, "left bdy position ="; Xbdy(0)
 PRINT #4, "type of left bdy ="; tlb;
 SELECT CASE tlb
    CASE 0
       PRINT #4, "  vacuum"
    CASE 0 TO 1
       PRINT #4, "  symmetric albedo {specular reflection factor}"
    CASE -1 TO 0
       PRINT #4, "  grey albedo {Lambertian reflection factor}"
    CASE ELSE
       PRINT #4, "  invalid"
 END SELECT
 PRINT #4, "current incident at left boundary ="; jinclb
 PRINT #4, "type of current incident at left bdy ="; tinclb;
 SELECT CASE tinclb
    CASE -1
       PRINT #4, "  Lambertian"
    CASE 0
       PRINT #4, "  isotropic surface Src"
    CASE 0 TO 1
       PRINT #4, "  abs(rmu) of collimated incident beam"
    CASE ELSE
       PRINT #4, "  invalid"
 END SELECT
 PRINT #4, "region #"; TAB(13); "cR"; TAB(23); "SigmaR"; TAB(37);
 PRINT #4, "SourceR"; TAB(49); "nc "; TAB(58); "Right Bdy"
 FOR ir = 1 TO nr
    PRINT #4, " "; ir; TAB(10);
    PRINT #4, USING "##.####"; cR(ir);
    PRINT #4, TAB(20);
    PRINT #4, USING "##.####^^^^"; SigmaR(ir);
    PRINT #4, TAB(34);
    PRINT #4, USING "##.####^^^^"; SourceR(ir);
    PRINT #4, TAB(48);
    PRINT #4, USING "###."; nc(ir);
    PRINT #4, TAB(58);
    PRINT #4, USING "###.####"; Xbdy(ir)
 NEXT ir
 PRINT #4, "type of right bdy ="; trb;
 SELECT CASE TypR
    CASE 0
       PRINT #4, "  vacuum"
    CASE 0 TO 1
       PRINT #4, "  symmetric albedo {specular reflection factor}"
```

```
   CASE -1 TO 0
      PRINT #4, "  grey albedo {Lambertian reflection factor}"
   CASE ELSE
      PRINT #4, "  invalid"
 END SELECT
 PRINT #4, "current incident at right boundary ="; jincrb
 PRINT #4, "type of current incident at right bdy ="; tincrb;
 SELECT CASE tincrb
   CASE -1
      PRINT #4, "  Lambertian"
   CASE 0
      PRINT #4, "  isotropic surface Src"
   CASE 0 TO 1
      PRINT #4, "  abs(rmu) of collimated incident beam"
   CASE ELSE
      PRINT #4, "  invalid"
 END SELECT

 IF ff% THEN PRINT #4, CHR$(12) ELSE PRINT #4, : PRINT #4,
 SELECT CASE DevEcho$
 CASE "LPT1", "SCREEN", "OUT"
   CLOSE #4
 END SELECT
RETURN

InputMethodParameters:
 SELECT CASE CmdProf$
  CASE "S"
   PRINT
   DO
      PRINT "Select Spatial Quadrature Method :"
      PRINT
      PRINT "    DD - Diamond Difference"
      PRINT "    LD - Linear Discontinuous"
      PRINT "    SC - Step Characteristic"
      PRINT "    LC - Linear Characteristic"
      PRINT "    LN - Linear Nodal"
      PRINT "    SA - Step   Adaptive"
      PRINT "    LA - Linear Adaptive"
      PRINT "    EC - Exponential Characteristic"
      PRINT
      INPUT "Choice? (DD,LD,SC,LC,LN,SA,LA,EC): ", DifMeth$
      DifMeth$ = UCASE$(LEFT$(LTRIM$(DifMeth$), 2))
   LOOP UNTIL INSTR("DDLDSCLCLNSALAEC", DifMeth$) AND LEN(DifMeth$) = 2
   SELECT CASE DifMeth$
      CASE "DD"
         PRINT "For Diamond Difference Spatial Quadrature:"
         fixup% = Choose("Use Negative Flux Fixup? (y/n): ")
         PRINT
      CASE "LN"
         PRINT "For Linear Nodal Spatial Quadrature:"
         fixup% = Choose("Use Scalar Flux Rotation Fixup? (y/n): ")
         PRINT
      CASE "LD", "SC", "LC", "SA", "LA"
      CASE "EC"
         PRINT
         CALL SetSwitchECtoLC
   END SELECT
   DO
      INPUT "Max number of iterations: IterMax = ", Itermax%
   LOOP UNTIL Itermax > 0
   PRINT
```

```
    PRINT "Criterion for convergence is that over an iteration,"
    PRINT "the relative change in each cell scalar flux <= Change"
    DO
       INPUT "Convergence Criterion: (0 < Change < 1)  Change = ",
Change
    LOOP UNTIL 0 < Change AND Change < 1
    PRINT
    DO
       INPUT "Tolerance for Moment Balance Comparisons (0< MomTol < 1)";
MomTol
    LOOP UNTIL Change > 0#
    PRINT
    DO
       PRINT "Select Angular Quadrature Method:"
       PRINT "   S - Single Range Gauss-Legendre"
       PRINT "   D - Double Range Gauss-Legendre"
       PRINT "   M - Composite Midpoint Rule"
       PRINT
       INPUT "Choice? (S/D/M): ", QuadMeth$
       QuadMeth$ = UCASE$(LEFT$(LTRIM$(QuadMeth$), 1))
    LOOP UNTIL INSTR("SDM", QuadMeth$) AND LEN(QuadMeth$) = 1
    PRINT
  CASE "A"
 END SELECT
 SELECT CASE QuadMeth$
    CASE "S"
       GOSUB SingleRangeGauss
    CASE "D"
       GOSUB DoubleRangeGauss
    CASE "M"
       GOSUB CompositeMidpointQuadrature
    CASE ELSE
       BEEP
       PRINT "ERROR: Illegal choice in InputMethodParams"
       STOP
 END SELECT
RETURN

SingleRangeGauss:
 SELECT CASE CmdProf$
  CASE "S"
    DO
       PRINT "For Single-Range Gauss Sn, n is total # of mu's."
       PRINT "Supported Orders are  n = 2, 4, 6, 8, 10 and 12."
       PRINT
       INPUT "Enter:  n = ", nk
    LOOP UNTIL (nk >= 2) AND (nk <= 12) AND (nk MOD 2 = 0)
    PRINT
  CASE "A"
 END SELECT
 REDIM w(nk), mu(nk)
 nk0 = nk
 GOSUB EvenGauss
RETURN

EvenGauss:
 SELECT CASE nk0
    CASE 2
       mu(1) = -.577350269189626#: w(1) = 1#
    CASE 4
       mu(1) = -.861136311594053#: w(1) = .347854845137454#
       mu(2) = -.339981043584856#: w(2) = .652145154862546#
```

```
    CASE 6
       mu(1) = -.9324695142031521#: w(1) = .17132449237917#
       mu(2) = -.661209386466265#: w(2) = .360761573048139#
       mu(3) = -.238619186083197#: w(3) = .467913934572691#
    CASE 8
       mu(1) = -.960289856497536#:  w(1) = .101228536290376#
       mu(2) = -.7966664774136269#: w(2) = .222381034453374#
       mu(3) = -.525532409916329#:  w(3) = .313706645877877#
       mu(4) = -.18343464249565#:   w(4) = .362683783378362#
    CASE 10
       mu(1) = -.973906528517172#: w(1) = .066671344308688#
       mu(2) = -.865063366688985#: w(2) = .149451349150581#
       mu(3) = -.679409568299024#: w(3) = .219086362515982#
       mu(4) = -.433395394129247#: w(4) = .269266719309996#
       mu(5) = -.148874338981631#: w(5) = .295524224714753#
    CASE 12
       mu(1) = -.981560634246719#: w(1) = .047175336386512#
       mu(2) = -.904117256370475#: w(2) = .106939325995318#
       mu(3) = -.769902674194305#: w(3) = .160078328543346#
       mu(4) = -.587317954286617#: w(4) = .203167426723066#
       mu(5) = -.36783149899818#: w(5) = .233492536538355#
       mu(6) = -.125233408511469#: w(6) = .249147045813403#
    CASE ELSE
       BEEP
       PRINT "ERROR: Gauss-Legendre with NK ="; nk0; " not supported."
       STOP
 END SELECT
 FOR k = nk0 TO nk0 \ 2 + 1 STEP -1
    mu(k) = -mu(nk0 - k + 1)
    w(k) = w(nk0 - k + 1)
 NEXT k
RETURN

DoubleRangeGauss:
 SELECT CASE CmdProf$
  CASE "S"
    DO
       PRINT "For Double-Range Gauss Sr., n is # of mu's in each range."
       PRINT "Supported Orders are  n = 1, 2, 3, 4, 6, 8, 10 and 12."
       INPUT "Enter:  N = ", nk
    LOOP UNTIL nk >= 1 AND nk <= 12 AND ((nk MOD 2 = 0) OR nk < 4)
   CASE "A"
  END SELECT
 nk0 = nk
 nk = 2 * nk0
 PRINT
 REDIM w(nk), mu(nk)
 SELECT CASE nk0
    CASE 1
       mu(1) = 0: w(1) = 2
    CASE 3
       mu(1) = -.7745966692#: w(1) = .5555555556#
       mu(2) = 0: w(2) = .8888888888000001#
       mu(3) = -mu(1): w(3) = w(1)
    CASE 2, 4, 6, 8, 10, 12
       GOSUB EvenGauss
    CASE ELSE
       BEEP
       PRINT "ERROR: Illegal nk0 in DoubleRangeGauss."
       STOP
 END SELECT
 'Shift and scale the above single-range quad set into the interval
```

```
(-1,0)
 FOR k = 1 TO nk0
    mu(k) = .5 * (mu(k) - 1): w(k) = .5 * w(k)
 NEXT k
 'mirror to the other single-range quad set, for interval (0,+1)
 FOR k = nk0 + 1 TO nk
    mu(k) = -mu(nk + 1 - k): w(k) = w(nk + 1 - k)
 NEXT k
RETURN

CompositeMidpointQuadrature:
 SELECT CASE CmdProf$
  CASE "S"
    DO
       PRINT "For Composite Midpoint Rule Sn, n is total # of mu's."
       PRINT "The mu's are at the center of equally sized intervals,"
       PRINT "    and are given equal weights."
       PRINT "Supported Orders are  n = 2, 4, 6, ..."
       INPUT "Enter:  N = ", nk
    LOOP UNTIL nk > 0 AND ((nk MOD 2) = 0)
    PRINT
  CASE "A"
 END SELECT
 REDIM w(nk), mu(nk)
 FOR k = 1 TO nk
    mu(k) = -1 + (2 * k - 1) / nk
    w(k) = 1 / nk
 NEXT k
RETURN

EchoMethod:
 SELECT CASE DevEcho$
    CASE "LPT1"
       OPEN "LPT1:" FOR OUTPUT AS #4
    CASE "SCREEN"
       OPEN "SCRN:" FOR OUTPUT AS #4
    CASE "OUT"
       OPEN Outfile$ FOR APPEND AS #4
 END SELECT
 SELECT CASE DifMeth$
    CASE "DD"
       a2$ = "Diamond Difference"
       IF fixup% THEN
          a3$ = " with Negative Flux Fixup."
       ELSE
          a3$ = " with NO Flux Fixup."
       END IF
    CASE "LD"
       a2$ = "Linear Discontinuous"
       a3$ = ""
    CASE "SC"
       a2$ = "Step Characteristic"
       a3$ = ""
    CASE "LC"
       a2$ = "Linear Characteristic"
       a3$ = ""
    CASE "LN"
       a2$ = "Linear Nodal"
       IF fixup% THEN
          a3$ = " with Scalar Flux Rotation Fixup."
       ELSE
          a3$ = " with NO Flux Rotation Fixup."
```

```
      END IF
    CASE "SA"
      a2$ = "Step Adaptive"
      a3$ = ""
    CASE "LA"
      a2$ = "Linear Adaptive"
      a3$ = ""
    CASE "EC"
      a2$ = "Exponential Characteristic"
      a3$ = ""
 END SELECT
 SELECT CASE CmdProf$
 CASE "S"
    PRINT #4, "************************************************************
******************"
    PRINT #4, StartTime$; "  "; StartDate$; "  "; V$
    PRINT #4, "Solution by Discrete Ordinates (Sn) with n = "; nk;
    PRINT #4, " mu's, total,"
    SELECT CASE QuadMeth$
      CASE "S"
        a1$ = "   with Single-Range Gauss-Legendre Angular Quadrature"
      CASE "D"
        a1$ = "   with Double-Range Gauss-Legendre Angular Quadrature"
      CASE "M"
        a1$ = "   with Composite Midpoint Angular Quadrature"
      CASE ELSE
        BEEP
        PRINT "ERROR: Unsupported angular quadrature (QuadMeth$) in
EchoMethod."
        STOP
    END SELECT
    PRINT #4, a1$
    PRINT #4, LTRIM$("and " + a2$ + " Spatial Quadtrature" + a3$ + ".")
    PRINT #4, "Convergence criterion on scalar fluxes: RelChange <
Change ="; Change
    PRINT #4,
  CASE "A"
   IF Eout$ = "FULL" THEN
    PRINT #4, "************************************************************
******************"
    PRINT #4, "Problem file    : "; UCASE$(Problemfile$); TAB(44); "nk
Angular Ordinates :"; nk
    PRINT #4, "Output file     : "; UCASE$(Outfile$); TAB(44);
"Quadrature Method    : "; UCASE$(QuadMeth$) + LTRIM$(STR$(nk))
    PRINT #4, "Tk file         : "; UCASE$(TKfile$); TAB(44); "Negative
Flux Fixups : "; UCASE$(Flfix$)
    PRINT #4, "Max Iterations  :"; Itermax%; TAB(44); "Reset Old Fluxes
: "; ResetAllFluxes$
    PRINT #4, "File Identifier : "; Ident$; TAB(44); "Solution Tolerance
:";
    PRINT #4, USING "##.##^^^^"; Change
    PRINT #4, "************************************************************
******************"
    PRINT #4, StartTime$; TAB(17); StartDate$; TAB(35); V$
    PRINT #4,
    IF DevEcho$ = "OUT" THEN PRINT #4, LTRIM$(a2$ + " -> ");
   ELSE
    PRINT #4, "************************************************************
******************"
    PRINT #4, StartTime$; TAB(17); StartDate$; TAB(35); V$
    PRINT #4,
    PRINT #4, LTRIM$(a2$ + " -> ");
```

```
   END IF
  END SELECT
  CLOSE #4
RETURN

InitializeProblem:
IF NewProblem% THEN
 nx = 0
 FOR ir = 1 TO nr
    nx = nx + nc(ir)
 NEXT ir
 REDIM Fa(nx, nk), Fx(nx, nk), J(nx, nk), Jinc(nk)
 REDIM FluxA(nx), FluxX(nx), Sigma(nx), c(nx), Source(nx)
 REDIM Sa(nx), Sx(nx), dx(nx), CellCtrX(nx)
 ix = 0
 FOR ir = 1 TO nr
    WidthX = (Xbdy(ir) - Xbdy(ir - 1)) / nc(ir)
    FOR ic = 1 TO nc(ir)
       ix = ix + 1
       Sigma(ix) = SigmaR(ir)
       c(ix) = cR(ir)
       Source(ix) = SourceR(ir)
       dx(ix) = WidthX
       CellCtrX(ix) = Xbdy(ir - 1) + (ic - .5) * WidthX
    NEXT ic
 NEXT ir
ELSE
 SELECT CASE DifMeth$
    CASE "DD"
       REDIM Fa(nx, nk), J(nx, nk), Jinc(nk)
    CASE "LD", "SC", "LC", "LN", "SA", "LA", "EC"
       REDIM Fa(nx, nk), Fx(nx, nk), J(nx, nk), Jinc(nk)
 END SELECT
END IF
IF jinclb = 0 THEN      'No incident current, so don't worry about type
 FOR k = nk TO nk / 2 + 1 STEP -1
    Jinc(k) = 0
 NEXT k
ELSE
 SELECT CASE tinclb
    CASE -1             'Lambertian incident current
       FOR k = nk TO nk / 2 + 1 STEP -1
          Jinc(k) = mu(k) * 4 * jinclb
       NEXT k
    CASE 0              'Isotropic Surface Source
       FOR k = nk TO nk / 2 + 1 STEP -1
          Jinc(k) = 2 * jinclb
       NEXT k
    CASE 0 TO 1         'Collimated Beam
       mu0 = tinclb
       k0 = nk + 1
       FOR k = nk TO nk / 2 + 1 STEP -1
          Jinc(k) = 0
          IF mu0 <= mu(k) THEN k0 = k
       NEXT k
       'k0 is index of smallest mu (in quadrature set) <= mu0
       IF k0 > nk THEN
          'mu0 > largest mu(k)
          Jinc(nk) = jinclb * (2 / w(nk))
       ELSEIF mu0 = mu(k0) THEN
          'mu0 = some mu(k)
          Jinc(k0) = jinclb * (2 / w(k0))
```

```
      ELSEIF k0 = nk / 2 + 1 THEN
         'mu0 < smallest pos. mu(k)
         Jinc(k0) = jinclb * (2 / w(k0))
      ELSE
         'mu(k0-1) < mu0 < mu(k0) so interpolate
jinc(k0) = jinclb * (mu0 - mu(k0 - 1)) / (mu(k0) - mu(k0 - 1))
Jinc(k0) = Jinc(k0) * (2 / w(k0)) * (mu(k0) / mu0)
Jinc(k0 - 1) = jinclb * (mu0 - mu(k0)) / (mu(k0 - 1) - mu(k0))
Jinc(k0 - 1) = Jinc(k0 - 1) * (2 / w(k0 - 1)) * (mu(k0 - 1) / mu0)
      END IF
 END SELECT
END IF
IF jincrb = 0 THEN      'No incident current, don't worry about type
 FOR k = 1 TO nk / 2
   Jinc(k) = 0
 NEXT k
ELSE
 SELECT CASE tincrb
   CASE -1      'Lambertian incident current
      FOR k = 1 TO nk / 2
         Jinc(k) = mu(k) * 4 * jincrb
      NEXT k
   CASE 0       'Isotropic Surface Source
      FOR k = 1 TO nk / 2
         Jinc(k) = -2 * jincrb
      NEXT k
   CASE 0 TO 1                    'Collimated Beam
      mu0 = -tincrb
      k0 = 0
      FOR k = 1 TO nk / 2
         Jinc(k) = 0
         IF mu0 >= mu(k) THEN k0 = k
      NEXT k
   'k0 is index of least negative mu (in quadrature set) <= mu0
      IF k0 < 1 THEN
         'mu0 < most neg. mu(k)
         Jinc(1) = -jincrb * (2 / w(1))
      ELSEIF mu0 = mu(k0) THEN
         'mu0 = some mu(k)
      Jinc(k0) = -jincrb * (2 / w(k0))
      ELSEIF k0 = nk / 2 THEN
         'mu0 > least neg. mu(k)
         Jinc(k0) = -jincrb * (2 / w(k0))
      ELSE
         'mu(k0) < mu0 < mu(k0+1) so interpolate
jinc(k0) = -jincrb * (2 / w(k0)) * (mu(k0) / mu0)
Jinc(k0) = Jinc(k0) * (mu0 - mu(k0 + 1)) / (mu(k0) - mu(k0 + 1))
Jinc(k0 + 1) = -jincrb * (2 / w(k0 + 1)) * (mu(k0 + 1) / mu0)
Jinc(k0 + 1) = Jinc(k0 + 1) * (mu0 - mu(k0)) / (mu(k0 + 1) - mu(k0))
      END IF
 END SELECT
END IF
RETURN

SnSolve:
 iter% = 0
 GOSUB UpdateSourceTermArrays
 DO
   DO
      iter% = iter% + 1
      GOSUB EnterLeftEdge
      GOSUB WalkRight
```

```
      GOSUB EnterRightEdge
      GOSUB WalkLeft
      GOSUB UpdateFluxesAndTestConvergence
      GOSUB UpdateSourceTermArrays
   LOOP UNTIL Converged% OR (iter% >= Itermax%)
   IF Converged% THEN
      EXIT DO
   ELSE
      BEEP
      PRINT "Not converged. MaxChangeObs ="; MaxChangeObs;
      PRINT " after"; iter%; "iterations."
      IF Choose("Perform additional iterations? (y/n): ") THEN
         DO
            INPUT "Additional number to do: ", IterAdditional%
         LOOP UNTIL IterAdditional% > 0
         Itermax% = Itermax% + IterAdditional%
      ELSE
         EXIT DO
      END IF
   END IF
 LOOP
RETURN

EnterLeftEdge:
 SELECT CASE tlb
   CASE 0                                          'Vacuum
      FOR k = nk TO nk / 2 + 1 STEP -1
         J(0, k) = Jinc(k)
      NEXT k
   CASE 1                                          'Symmetry
      FOR k = nk TO nk / 2 + 1 STEP -1
         J(0, k) = Jinc(k) - J(0, nk - k + 1)
      NEXT k
   CASE 0 TO 1                                     'Specular Albedo
      FOR k = nk TO nk / 2 + 1 STEP -1
         J(0, k) = Jinc(k) - tlb * J(0, nk - k + 1)
      NEXT k
   CASE -1 TO 0                                    'Grey Albedo
      Jminus = 0
      FOR k = 1 TO nk / 2
         Jminus = Jminus - J(0, k) * w(k)
      NEXT k
      Jminus = Jminus * .5
      Fplus = Jminus * 4 * ABS(tlb)
      FOR k = nk TO nk / 2 + 1 STEP -1
         J(0, k) = Jinc(k) + Fplus * mu(k)
      NEXT k
 END SELECT
RETURN

WalkRight:
 'lprint "Starting WalkRight, iter% =";iter%
 SELECT CASE DifMeth$
 CASE "DD"
   FOR k = nk TO nk / 2 + 1 STEP -1
      FOR ix = 1 TO nx
         StepDD J(ix - 1, k), J(ix, k), Sa(ix), dx(ix), Sigma(ix),
mu(k), Fa(ix, k)
      NEXT ix
   NEXT k
 CASE "LD"
   'LD uses same Step algorithm as LN, but using a Pade expansion
```

```
    'for exp(-eps) in MakeP converts implicitly to LD
    'Note this method involves NO Rotational Fixup
    FOR k = nk TO nk / 2 + 1 STEP -1
       FOR ix = 1 TO nx
          Fin = J(ix - 1, k) / mu(k)
          StepLN Sigma(ix), dx(ix), mu(k), Fin, Sa(ix), Sx(ix), Fout,
Fa(ix, k), Fx(ix, k)
          J(ix, k) = Fout * mu(k)
       NEXT ix
    NEXT k
 CASE "SC"
    FOR k = nk TO nk / 2 + 1 STEP -1
       FOR ix = 1 TO nx
          Fin = J(ix - 1, k) / mu(k)
          StepSC Sigma(ix), dx(ix), mu(k), Fin, Sa(ix), Sx(ix), Fout,
Fa(ix, k), Fx(ix, k)
          J(ix, k) = Fout * mu(k)
       NEXT ix
    NEXT k
 CASE "LC"
    FOR k = nk TO nk / 2 + 1 STEP -1
       FOR ix = 1 TO nx
          Fin = J(ix - 1, k) / mu(k)
          StepLC Sigma(ix), dx(ix), mu(k), Fin, Sa(ix), Sx(ix), Fout,
Fa(ix, k), Fx(ix, k)
          J(ix, k) = Fout * mu(k)
       NEXT ix
    NEXT k
 CASE "LN"
    FOR k = nk TO nk / 2 + 1 STEP -1
       FOR ix = 1 TO nx
          Fin = J(ix - 1, k) / mu(k)
          StepLN Sigma(ix), dx(ix), mu(k), Fin, Sa(ix), Sx(ix), Fout,
Fa(ix, k), Fx(ix, k)
          J(ix, k) = Fout * mu(k)
       NEXT ix
    NEXT k
 CASE "SA"
    FOR k = nk TO nk / 2 + 1 STEP -1
       FOR ix = 1 TO nx
          Fin = J(ix - 1, k) / mu(k)
          StepSA Sigma(ix), dx(ix), mu(k), Fin, Sa(ix), Sx(ix), Fout,
Fa(ix, k), Fx(ix, k)
          J(ix, k) = Fout * mu(k)
       NEXT ix
    NEXT k
 CASE "LA"
    FOR k = nk TO nk / 2 + 1 STEP -1
       FOR ix = 1 TO nx
          Fin = J(ix - 1, k) / mu(k)
          StepLA Sigma(ix), dx(ix), mu(k), Fin, Sa(ix), Sx(ix), Fout,
Fa(ix, k), Fx(ix, k)
          J(ix, k) = Fout * mu(k)
          'lprint j(ix-1,k), j(ix,k), Sa(ix)
          'lprint (sigma(ix) * dx(ix) * 0.5 / abs(mu(k))), mu(k),
Fa(ix,k)
       NEXT ix
    NEXT k
 CASE "EC"
    FOR k = nk TO nk / 2 + 1 STEP -1
       FOR ix = 1 TO nx
          Fin = J(ix - 1, k) / mu(k)
```

```
         StepEC Sigma(ix), dx(ix), mu(k), Fin, Sa(ix), Sx(ix), Fout,
Fa(ix, k), Fx(ix, k)
         J(ix, k) = Fout * mu(k)
      NEXT ix
   NEXT k
 END SELECT
RETURN

EnterRightEdge:
 SELECT CASE trb
   CASE 0                                          'Vacuum
      FOR k = 1 TO nk / 2
         J(nx, k) = Jinc(k)
      NEXT k
   CASE 1                                          'Symmetry
      FOR k = 1 TO nk / 2
         J(nx, k) = Jinc(k) - J(nx, nk - k + 1)
      NEXT k
   CASE 0 TO 1                                     'Specular Albedo
      FOR k = 1 TO nk / 2
         J(nx, k) = Jinc(k) - trb * J(nx, nk - k + 1)
      NEXT k
   CASE -1 TO 0                                    'Grey Albedo
      Jplus = 0
      FOR k = nk TO nk / 2 + 1 STEP -1
         Jplus = Jplus + J(nx, k) * w(k)
      NEXT k
      Jplus = Jplus * .5
      Fminus = Jplus * 4 * ABS(trb)
      FOR k = 1 TO nk / 2
         J(nx, k) = Jinc(k) + Fminus * mu(k)
      NEXT k
 END SELECT
RETURN

WalkLeft:
 'lprint "Starting WalkLeft, iter% =";iter%
 SELECT CASE DifMeth$
 CASE "DD"
   FOR k = 1 TO nk / 2
      FOR ix = nx TO 1 STEP -1
         StepDD J(ix, k), J(ix - 1, k), Sa(ix), dx(ix), Sigma(ix),
mu(k), Fa(ix, k)
      NEXT ix
   NEXT k
 CASE "LD"
   'LD uses same Step algorithm as LN, but using a Pade expansion
   'for exp(-eps) in MakeP converts implicitly to LD
   'Note this method involves NO Rotational Fixup
   FOR k = 1 TO nk / 2
      FOR ix = nx TO 1 STEP -1
         Fin = J(ix, k) / mu(k)
         StepLN Sigma(ix), dx(ix), -mu(k), Fin, Sa(ix), -Sx(ix), Fout,
Fa(ix, k), Fx(ix, k)
   'Walk leftward, so reflect cell, hence, use negative of mu and Sx
         Fx(ix, k) = -Fx(ix, k) 'and also get back negative of Fx
         J(ix - 1, k) = Fout * mu(k)
      NEXT ix
   NEXT k
 CASE "SC"
   FOR k = 1 TO nk / 2
      FOR ix = nx TO 1 STEP -1
```

```
         Fin = J(ix, k) / mu(k)
         StepSC Sigma(ix), dx(ix), -mu(k), Fin, Sa(ix), -Sx(ix), Fout,
Fa(ix, k), Fx(ix, k)
    'Walk leftward, so reflect cell, hence, use negative of mu and Sx
         J(ix - 1, k) = Fout * mu(k)
      NEXT ix
   NEXT k
 CASE "LC"
   FOR k = 1 TO nk / 2
      FOR ix = nx TO 1 STEP -1
         Fin = J(ix, k) / mu(k)
         StepLC Sigma(ix), dx(ix), -mu(k), Fin, Sa(ix), -Sx(ix), Fout,
Fa(ix, k), Fx(ix, k)
    'Walk leftward, so reflect cell, hence, use negative of mu and Sx
         Fx(ix, k) = -Fx(ix, k) 'and also get back negative of Fx
         J(ix - 1, k) = Fout * mu(k)
      NEXT ix
   NEXT k
 CASE "LN"
   FOR k = 1 TO nk / 2
      FOR ix = nx TO 1 STEP -1
         Fin = J(ix, k) / mu(k)
         StepLN Sigma(ix), dx(ix), -mu(k), Fin, Sa(ix), -Sx(ix), Fout,
Fa(ix, k), Fx(ix, k)
    'Walk leftward, so reflect cell, hence, use negative of mu and Sx
         Fx(ix, k) = -Fx(ix, k) 'and also get back negative of Fx
         J(ix - 1, k) = Fout * mu(k)
      NEXT ix
   NEXT k
 CASE "SA"
   FOR k = 1 TO nk / 2
      FOR ix = nx TO 1 STEP -1
         Fin = J(ix, k) / mu(k)
         StepSA Sigma(ix), dx(ix), -mu(k), Fin, Sa(ix), -Sx(ix), Fout,
Fa(ix, k), Fx(ix, k)
    'Walk leftward, so reflect cell, hence, use negative of mu and Sx
         Fx(ix, k) = -Fx(ix, k) 'and also get back negative of Fx
         J(ix - 1, k) = Fout * mu(k)
      NEXT ix
   NEXT k
 CASE "LA"
   FOR k = 1 TO nk / 2
      FOR ix = nx TO 1 STEP -1
         Fin = J(ix, k) / mu(k)
         StepLA Sigma(ix), dx(ix), -mu(k), Fin, Sa(ix), -Sx(ix), Fout,
Fa(ix, k), Fx(ix, k)
    'Walk leftward, so reflect cell, hence, use negative of mu and Sx
         Fx(ix, k) = -Fx(ix, k) 'and also get back negative of Fx
         J(ix - 1, k) = Fout * mu(k)
         'lprint j(ix-1,k), j(ix,k), Sa(ix)
         'lprint (sigma(ix) * dx(ix) * 0.5 / abs(mu(k))), mu(k),
Fa(ix,k)
      NEXT ix
   NEXT k
 CASE "EC"
   FOR k = 1 TO nk / 2
      FOR ix = nx TO 1 STEP -1
         Fin = J(ix, k) / mu(k)
         StepEC Sigma(ix), dx(ix), -mu(k), Fin, Sa(ix), -Sx(ix), Fout,
Fa(ix, k), Fx(ix, k)
    'Walk leftward, so reflect cell, hence, use negative of mu and Sx
         Fx(ix, k) = -Fx(ix, k) 'and also get back negative of Fx
```

```
            J(ix - 1, k) = Fout * mu(k)
        NEXT ix
    NEXT k
 END SELECT
RETURN

UpdateFluxesAndTestConvergence:
MaxChangeObs = 0
SELECT CASE DifMeth$
CASE "DD"
 Sum = 0
 FOR k = 1 TO nk
    Sum = Sum + w(k) * J(0, k) / mu(k)
 NEXT k
 BdyFlux = Sum / 2
 FOR ix = 1 TO nx
    Sum = 0
    FOR k = 1 TO nk
       Sum = Sum + w(k) * Fa(ix, k)
    NEXT k
    FluxNewA = Sum / 2
    IF (ABS(FluxNewA) <= Change) AND (ABS(FluxA(ix)) <= Change) THEN
       ChangeObs = MAX(ABS(FluxNewA), ABS(FluxA(ix)))
    ELSE
       ChangeObs = ABS(FluxNewA - FluxA(ix)) * 2 / (FluxNewA +
FluxA(ix))
    END IF
    MaxChangeObs = MAX(MaxChangeObs, ChangeObs)
    FluxA(ix) = FluxNewA
    Sum = 0
    FOR k = 1 TO nk
       Sum = Sum + w(k) * J(ix, k) / mu(k)
    NEXT k
    FluxX(ix) = (Sum / 2) - BdyFlux
    BdyFlux = Sum / 2
 NEXT ix
CASE "LN"
 FOR ix = 1 TO nx
    SumA = 0
    SumX = 0
    FOR k = 1 TO nk
       SumA = SumA + w(k) * Fa(ix, k)
       SumX = SumX + w(k) * Fx(ix, k)
    NEXT k
    FluxNewA = SumA / 2
    FluxNewX = SumX / 2
    IF (ABS(FluxNewA) <= Change) AND (ABS(FluxA(ix)) <= Change) THEN
      ChangeObs = MAX(ABS(FluxNewA), ABS(FluxA(ix)))
    ELSE
       ChangeObs = ABS(FluxNewA - FluxA(ix)) * 2 / (FluxNewA +
FluxA(ix))
    END IF
    MaxChangeObs = MAX(MaxChangeObs, ChangeObs)
    FluxA(ix) = FluxNewA
    FluxX(ix) = FluxNewX
    IF fixup% AND ABS(FluxX(ix)) > FluxA(ix) THEN
       FluxX(ix) = SGN(FluxX(ix)) * FluxA(ix)
    END IF
 NEXT ix
CASE "LD", "SC", "LC", "SA", "LA", "EC"
 FOR ix = 1 TO nx
    SumA = 0
```

```
    SumX = 0
    FOR k = 1 TO nk
       SumA = SumA + w(k) * Fa(ix, k)
       SumX = SumX + w(k) * Fx(ix, k)
    NEXT k
    FluxNewA = SumA / 2
    FluxNewX = SumX / 2
    IF (ABS(FluxNewA) <= Change) AND (ABS(FluxA(ix)) <= Change) THEN
      ChangeObs = MAX(ABS(FluxNewA), ABS(FluxA(ix)))
    ELSE
       ChangeObs = ABS(FluxNewA - FluxA(ix)) * 2 / (FluxNewA +
FluxA(ix))
    END IF
    MaxChangeObs = MAX(MaxChangeObs, ChangeObs)
    FluxA(ix) = FluxNewA
    FluxX(ix) = FluxNewX
 NEXT ix
END SELECT
  LOCATE 15, 1
  PRINT " Working "; DifMeth$; TAB(15);
  timiters = (TIMER - StartClock) / iter% / 60
  PRINT USING "--> ####.####"; timiters;
  PRINT " min/iter ";
  IF (iter% = 1) AND (iter1% > 1) THEN
    LOCATE 16, 15
    timiters = timiters * iter1%
    PRINT USING "Est ####.####"; timiters;
    PRINT " min to go "
  END IF
  LOCATE 18, 1
  PRINT " After Iteration"; iter1%; "  MaxChangeObs ="; MaxChangeObs1; "
"
  PRINT "                 "; iter%; "                ="; MaxChangeObs; "
"
  iter1% = iter%
  MaxChangeObs1 = MaxChangeObs
  PRINT
  Converged% = (MaxChangeObs < Change)
RETURN

UpdateSourceTermArrays:
 SELECT CASE DifMeth$
 CASE "DD"
    FOR ix = 1 TO nx
       Sa(ix) = c(ix) * Sigma(ix) * FluxA(ix) + Source(ix)
    NEXT ix
 CASE "LC"
    FOR ix = 1 TO nx
       Sa(ix) = c(ix) * Sigma(ix) * FluxA(ix) + Source(ix)
       Sx(ix) = c(ix) * Sigma(ix) * FluxX(ix)
       'Implement Negative Source Fixup as required for LC
       IF ABS(Sx(ix)) > Sa(ix) THEN
         Sx(ix) = SGN(Sx(ix)) * Sa(ix)
       END IF
    NEXT ix
 CASE "LD", "LN", "SC", "SA", "LA", "EC"
    FOR ix = 1 TO nx
       Sa(ix) = c(ix) * Sigma(ix) * FluxA(ix) + Source(ix)
       Sx(ix) = c(ix) * Sigma(ix) * FluxX(ix)
    NEXT ix
 END SELECT
RETURN
```

```
CalculateResults:
 REDIM JPlusBdy(nr), JMinusBdy(nr), JNetBdy(nr), FluxBdy(nr)
 REDIM FluxAveR(nr), FluxXR(nr)
 ix = 0
 ir = 0
 GOSUB CalculateRegionBoundaryResults
 'left bdy of first region
 FOR ir = 1 TO nr
    GOSUB CalculateRegionAverageScalarFlux
    'ir'th region
    GOSUB CalculateRegionBoundaryResults
    'right bdy of ir'th region
 NEXT ir
RETURN

CalculateRegionBoundaryResults:
 Sum = 0
 FOR k = nk TO nk / 2 + 1 STEP -1
    Sum = Sum + w(k) * J(ix, k)
 NEXT k
 JPlusBdy(ir) = Sum / 2
 Sum = 0
 FOR k = 1 TO nk / 2
    Sum = Sum + w(k) * J(ix, k)
 NEXT k
 JMinusBdy(ir) = -Sum / 2
 JNetBdy(ir) = JPlusBdy(ir) - JMinusBdy(ir)
 Sum = 0
 FOR k = 1 TO nk
    Sum = Sum + w(k) * J(ix, k) / mu(k)
 NEXT k
 FluxBdy(ir) = Sum / 2
RETURN

CalculateRegionAverageScalarFlux:
 SumA = 0
 SumX = 0
 Sum1 = 0
 FOR ic = 1 TO nc(ir)
    ix = ix + 1
    SumA = SumA + FluxA(ix)
    SumX = SumX + FluxX(ix)
    Sum1 = Sum1 + (2 * ic - 1) * FluxA(ix)
 NEXT ic
 FluxAveR(ir) = SumA / nc(ir)
 FluxXR(ir) = -3 * SumA / nc(ir) + (SumX + 3 * Sum1) / nc(ir) ^ 2
RETURN

VerifyRegionsByBalanceEquation:
 MaxRegrerr = 0!
 RegBalFlag$ = ""
 Regflag% = False
 FOR ir = 1 TO nr
    SumBL = (JNetBdy(ir) - JNetBdy(ir - 1)) / (Xbdy(ir) - Xbdy(ir - 1))
+ SigmaR(ir) * FluxAveR(ir)
    SumBR = (cR(ir) * SigmaR(ir) * FluxAveR(ir) + SourceR(ir))
    IF (SumBL <> 0) OR (SumBR <> 0) THEN
       Regrerr = ABS(SumBL - SumBR) * 2 / (ABS(SumBL) + ABS(SumBR))
    ELSE
       Regrerr = 0
    END IF
    IF Regrerr < 2 THEN MaxRegrerr = MAX(MaxRegrerr, Regrerr)
```

```
    IF MaxRegrerr > Change THEN
       Regflag% = True
       RegBalFlag$ = LTRIM$(RegBalFlag$ + STR$(ir))
    END IF
 NEXT ir
RETURN

LPrintRegionSummary:
 SELECT CASE DevEcho$
    CASE "LPT1"
       OPEN "LPT1:" FOR OUTPUT AS #4
    CASE "SCREEN"
       OPEN "SCRN:" FOR OUTPUT AS #4
    CASE "OUT"
       OPEN Outfile$ FOR APPEND AS #4
 END SELECT
 SELECT CASE DifMeth$
    CASE "DD"
       IF fixup% THEN
          PRINT #4, "DDF Negative Flux Fixup Enabled"
       ELSE
          PRINT #4, "DD No Flux Fixup"
       END IF
    CASE "LC"
       PRINT #4, "LC Source Rotation (Sx<=Sa) Fixup ALWAYS Enabled"
    CASE "LN"
       IF fixup% THEN
          PRINT #4, "LNF Scalar Flux Rotation (PhiX<PhiA) Fixup Enabled"
       ELSE
          PRINT #4, "LN No Flux Fixup"
       END IF
    CASE "EC"
       IF SwitchEC% THEN
          PRINT #4, "EC to LC Enabled  (beta <=";
       ELSE
          PRINT #4, "EC Normal         (beta > ";
       END IF
       PRINT #4, USING "##.####^^^^"; SwECtoLCSetPt;
       PRINT #4, ")"
    CASE ELSE
       PRINT #4, DifMeth$
 END SELECT
 PRINT #4, "Execution Time : ";
 PRINT #4, USING "###.####"; ExecMin;
 PRINT #4, " min "
 IF Converged% THEN PRINT #4, "Converged ";
 PRINT #4, "After"; iter%; "iterations, MaxChangeObs ="; MaxChangeObs
 PRINT #4,
 PRINT #4, "", " J plus", "  J minus", "   J net", "    Boundary Flux"
 PRINT #4, "Region #", " flux ave", "  flux x-moment"
 PRINT #4,
 ir = 0
 PRINT #4, "",
 PRINT #4, USING " +#.######^^^^ "; JPlusBdy(ir); JMinusBdy(ir);
 PRINT #4, USING " +#.######^^^^ "; JNetBdy(ir); FluxBdy(ir)
 FOR ir = 1 TO nr
    PRINT #4, ir,
    PRINT #4, USING " +#.######^^^^ "; FluxAveR(ir); FluxXR(ir)
    PRINT #4, "",
    PRINT #4, USING " +#.######^^^^ "; JPlusBdy(ir); JMinusBdy(ir);
    PRINT #4, USING " +#.######^^^^ "; JNetBdy(ir); FluxBdy(ir)
 NEXT ir
```

```
 IF (MaxMom0rerr > MomTol) THEN
   Mom0IntFlag$ = " 0th: VIOLATION"
 END IF
 IF (MaxMom1rerr > MomTol) THEN
   Mom1IntFlag$ = " 1st: VIOLATION"
 END IF
 PRINT #4,
 PRINT #4, "Moment Balances: "; Mom0IntFlag$;
 PRINT #4, USING "    MaxRelErr: ##.####^^^^"; MaxMom0rerr;
 PRINT #4, "  ≤ ";
 PRINT #4, USING "##.####^^^^"; MomTol
 SELECT CASE DifMeth$
    CASE "DD", "SC"
    CASE ELSE
       PRINT #4, TAB(18); Mom1IntFlag$;
       PRINT #4, USING "                ##.####^^^^"; MaxMom1rerr;
       PRINT #4, "  ≤ ";
       PRINT #4, USING "##.####^^^^"; MomTol
 END SELECT
 PRINT #4, "      Region Balances: ";
 IF Regflag% = True THEN
    PRINT #4, TAB(24); LTRIM$("VIOLATED in " + RegBalFlag$ + ": ");
 ELSE
    PRINT #4, "CONSERVED"; TAB(48);
 END IF
 PRINT #4, USING "##.####^^^^"; MaxRegrerr;
 PRINT #4, "  ≤ ";
 PRINT #4, USING "##.####^^^^"; Change
 PRINT #4, : PRINT #4,
 CLOSE #4
RETURN

RunSummary:
 SELECT CASE DevEcho$
    CASE "LPT1"
       OPEN "LPT1:" FOR OUTPUT AS #4
    CASE "SCREEN"
       OPEN "SCRN:" FOR OUTPUT AS #4
    CASE "OUT"
       OPEN Outfile$ FOR APPEND AS #4
 END SELECT
 MinChangeReg = 1E+15
 ilotogl = -2: ihitogl = 0
 FOR it = 1 TO itim
    GOSUB ToggleDifMeth
    IF DifMeth$ = "EC" THEN iec = it
    MinChangeReg = MIN(MinChangeReg, RErranal(it))
    RelRtim(it) = (1000! * Rtim(it))
    RelRErranal(it) = (1000! * RErranal(it))
 NEXT it
 IF iec < 1 OR iec > itim THEN iec = 1
 PRINT #4, "*****************************************************************
***************"
 PRINT #4,
 PRINT #4, "   RunSummary: Relative Processing times and Region
MaxChanges"
 PRINT #4, "     Method   Exec Min    RelTime     RegMaxErr
RelRegErr"
 PRINT #4,
 ilotogl = -2: ihitogl = 0
 FOR it = 1 TO itim
    GOSUB ToggleDifMeth
```

```
    IF (DifMeth$ = "DD" OR DifMeth$ = "LN") AND (fixup%) THEN
       fixa$ = "F"
    ELSE fixa$ = ""
    END IF
    PRINT #4, TAB(8); LTRIM$(DifMeth$ + fixa$); TAB(14);
    PRINT #4, USING "####.####"; Rtim(it);
    PRINT #4, TAB(26);
    PRINT #4, USING "###.####"; CSNG(RelRtim(it) / RelRtim(iec));
    PRINT #4, TAB(37);
    PRINT #4, USING "##.####^^^^"; RErranal(it);
    PRINT #4, TAB(51);
    IF MinChangeReg <> 0 THEN
       PRINT #4, USING "###.####"; CSNG(RelRErranal(it) / (1000 *
MinChangeReg))
    ELSE PRINT #4, "--------"
    END IF
 NEXT it
 PRINT #4, : PRINT #4,
 CLOSE #4
RETURN

CreateTKDataFile:
 SELECT CASE CmdProf$
    CASE "S"
       DO
          INPUT "Output Data to a TK File? ( Y/N ): ", Qtkf$
          Qtkf$ = UCASE$(LEFT$(LTRIM$(Qtkf$), 1))
       LOOP UNTIL INSTR("YN", Qtkf$) AND LEN(Qtkf$) = 1
    CASE "A"
 END SELECT
 IF UCASE$(Qtkf$) = "Y" THEN
    SELECT CASE CmdProf$
       CASE "S"
          DO
             INPUT "TK File name? : ", TKfile$
          LOOP WHILE TKfile$ <> ""
          OPEN TKfile$ FOR APPEND AS #3
       CASE "A"
          OPEN TKfile$ FOR APPEND AS #3
    END SELECT
    IF (DifMeth$ = "DD" OR DifMeth$ = "LN") AND (fixup%) THEN
       fixa$ = "F"
    ELSE fixa$ = ""
    END IF
    CALL PrtTK(LTRIM$("cR_" + DifMeth$ + fixa$), cR(), 1, nr)
    CALL PrtTK(LTRIM$("SigR_" + DifMeth$ + fixa$), SigmaR(), 1, nr)
    CALL PrtTK(LTRIM$("Src_" + DifMeth$ + fixa$), SourceR(), 1, nr)
    REDIM cellsnc(1 TO nr)
    FOR ir = 1 TO nr
       cellsnc(ir) = nc(ir)
    NEXT ir
    CALL PrtTK(LTRIM$("nc_" + DifMeth$ + fixa$), cellsnc(), 1, nr)
    CALL PrtTK(LTRIM$("Xb_" + DifMeth$ + fixa$), Xbdy(), 0, nr)
    CALL PrtTK(LTRIM$("Jposb_" + DifMeth$ + fixa$), JPlusBdy(), 0, nr)
    CALL PrtTK(LTRIM$("Jnegb_" + DifMeth$ + fixa$), JMinusBdy(), 0, nr)
    CALL PrtTK(LTRIM$("Jnetb_" + DifMeth$ + fixa$), JNetBdy(), 0, nr)
    CALL PrtTK(LTRIM$("Phib_" + DifMeth$ + fixa$), FluxBdy(), 0, nr)
    REDIM Xctr(nr)
    FOR ir = 1 TO nr
       Xctr(ir) = Xbdy(ir - 1) + (Xbdy(ir) - Xbdy(ir - 1)) / 2!
    NEXT ir
    CALL PrtTK(LTRIM$("Xctr_" + DifMeth$ + fixa$), Xctr(), 1, nr)
```

```
    CALL PrtTK(LTRIM$("PhiA_" + DifMeth$ + fixa$), FluxAveR(), 1, nr)
    CALL PrtTK(LTRIM$("PhiX_" + DifMeth$ + fixa$), FluxXR(), 1, nr)
    'establish a global flux variable
    ng = 2 * nr
    REDIM XGlbl(0 TO ng), FlxGlbl(0 TO ng)
    ig = 0: ix = 0: iy = 1
    DO
       XGlbl(ig) = Xbdy(ix)
       FlxGlbl(ig) = FluxBdy(ix)
       ix = ix + 1
       ig = ig + 1
       IF ix <> (nr + 1) THEN
          XGlbl(ig) = Xctr(iy)
          FlxGlbl(ig) = FluxAveR(iy)
          iy = iy + 1
          ig = ig + 1
       END IF
    LOOP UNTIL ix = (nr + 1)
    CALL PrtTK(LTRIM$("XG_" + DifMeth$ + fixa$), XGlbl(), 0, ng)
    CALL PrtTK(LTRIM$("PhiG_" + DifMeth$ + fixa$), FlxGlbl(), 0, ng)
    CALL PrtTK(LTRIM$("Time_" + DifMeth$ + fixa$), Rtim(), itim, itim)
    PRINT #3,
    CLOSE #3
 END IF
RETURN

SUB AltMenu STATIC
SHARED Problemfile$, Outfile$, Qoutf$, TKfile$, Qtkf$, DifMeth$,
TotDifMeth$
SHARED ilotogl, ihitogl, QuadMeth$, fixup%, Flfix$, Itermax%, Change
SHARED MomTol, nk, Ident$, again%, NewProblem%, ResetAllFluxes$
SHARED SwECtoLCSetPt
Variables:
CLS
PRINT "*************************************************************
*********"
PRINT "Problem File      : "; UCASE$(Problemfile$); TAB(40); "nk Angular
Ordinates :"; nk
PRINT "Output file       : "; UCASE$(Outfile$); TAB(40); "Quadrature
Method    : "; UCASE$(QuadMeth$) + LTRIM$(STR$(nk))
PRINT "Tk file           : "; UCASE$(TKfile$); TAB(40); "Negative Flux
Fixups : "; UCASE$(Flfix$)
PRINT "Max Iterations    :"; Itermax%; TAB(40); "Reset All Old Fluxes :
"; ResetAllFluxes$
PRINT "Moment Tolerance  :";
PRINT USING "##.##^^^^"; MomTol;
PRINT TAB(40); "Solution Tolerance   :";
PRINT USING "##.##^^^^"; Change
PRINT
PRINT "        Spatial Quadrature Methods :"; TotDifMeth$
PRINT "                    File Identifier : "; Ident$
PRINT "*************************************************************
*********"
PRINT
PRINT "Type   to change "; TAB(40); "Type   to change "
PRINT "-----  ---------------------------"; TAB(40); "-----
---------------------------"
IF NOT again% THEN
 PRINT " PF    Problem File  "; TAB(40); " NK    # Angular Ordinates "
 PRINT " O     Data Output File"; TAB(40); " QM    Quadrature Method"
 PRINT " TK    TK Solver Output File"; TAB(40); " FF    Neg Flux Fixups
"
```

```
ELSE
 PRINT " --     Problem File  "; TAB(40); " NK     # Angular Ordinates "
 PRINT " --     Data Output File"; TAB(40); " QM     Quadrature Method"
 PRINT " --     TK Solver Output File"; TAB(40); " FF     Neg Flux Fixups
"
END IF
PRINT " MI     Max Iterations Allowed "; TAB(40); " TL     Change
Tolerances"
PRINT " AS     Add   Spatial Quad Method "; TAB(40); " NF     Reset Flux
Usage Toggle"
PRINT " CS     Clear Spatial Quad Methods "; TAB(40); " FL     DOS File
Directory"
PRINT "  N     None of the above"
PRINT
INPUT Response$
 LOCATE 12, 1
 FOR i = 1 TO 12
    PRINT TAB(79); " "
 NEXT i
 LOCATE 12, 1
IF UCASE$(Response$) = "FL" AND NOT again% THEN
 INPUT "Path or file: "; fileinq$
 FILES (fileinq$)
 PRINT : PRINT "Holding 5 Seconds... ": PRINT
 SLEEP (5)
ELSEIF UCASE$(Response$) = "PF" AND NOT again% THEN
 INPUT "Problem File : "; Problemfile$
ELSEIF UCASE$(Response$) = "O" AND NOT again% THEN
 INPUT "Output file (OFF for none) : "; Outfile$
 IF UCASE$(Outfile$) = "OFF" THEN
    Qoutf$ = "N"
  ELSE
    Qoutf$ = "Y"
    Qtkf$ = "Y"
    ippos = 0
    ippos = INSTR(Outfile$, ".") - 1
    TKfile$ = LTRIM$(MID$(Outfile$, 1, ippos) + ".TKD")
 END IF
ELSEIF UCASE$(Response$) = "TK" AND NOT again% THEN
 INPUT "Tk file  (OFF for none) : "; TKfile$
 Qtkf$ = "Y"
 IF UCASE$(TKfile$) = "OFF" THEN
    Qtkf$ = "N"
 END IF
ELSEIF UCASE$(Response$) = "MI" THEN
 DO
    INPUT "Maximum Allowable Iterations "; Itermax%
 LOOP UNTIL Itermax% > 1
ELSEIF UCASE$(Response$) = "NK" THEN
 SELECT CASE QuadMeth$
 CASE "S"
    DO
       PRINT "For Single-Range Gauss Sn, n is total # of mu's."
       PRINT "Supported Orders are  n = 2, 4, 6, 8, 10 and 12."
       PRINT
       INPUT "Enter:  n = ", nk
    LOOP UNTIL (nk >= 2) AND (nk <= 12) AND (nk MOD 2 = 0)
 CASE "D"
    DO
       PRINT "For Double-Range Gauss Sn, n is # of mu's in each range."
       PRINT "Supported Orders are  n = 1, 2, 3, 4, 6, 8, 10 and 12."
       PRINT
```

```
      INPUT "Enter:  N = ", nk
   LOOP UNTIL nk >= 1 AND nk <= 12 AND ((nk MOD 2 = 0) OR nk < 4)
 CASE "M"
   DO
      PRINT "For Composite Midpoint Rule Sn, n is total # of mu's."
      PRINT "The mu's are at the center of equally sized intervals,"
      PRINT "    and are given equal weights."
      PRINT "Supported Orders are  n = 2, 4, 6, ..."
      PRINT
      INPUT "Enter:  N = ", nk
   LOOP UNTIL nk > 0 AND ((nk MOD 2) = 0)
 END SELECT
ELSEIF UCASE$(Response$) = "QM" THEN
 DO
   PRINT "Select Angular Quadrature Type:": PRINT
   PRINT "   S - Single Range Gauss-Legendre"
   PRINT "   D - Double Range Gauss-Legendre"
   PRINT "   M - Composite Midpoint Rule": PRINT
   INPUT "Choice? ( S,D,M ): ", QuadMeth$
   QuadMeth$ = UCASE$(LEFT$(LTRIM$(QuadMeth$), 1))
 LOOP UNTIL INSTR("SDM", QuadMeth$) AND LEN(QuadMeth$) = 1
ELSEIF UCASE$(Response$) = "FF" THEN
 DO
   INPUT "Provide Negative Flux Fixups (Enter Y or N only) "; Flfix$
 LOOP UNTIL (UCASE$(Flfix$) = "Y") OR (UCASE$(Flfix$) = "N")
 IF UCASE$(Flfix$) = "Y" THEN
   Flfix$ = "YES": fixup% = True
  ELSE
   Flfix$ = "NO": fixup% = False
  END IF
ELSEIF UCASE$(Response$) = "TL" THEN
 DO
   INPUT "Tolerance for Solution Convergence (0 to 1) : "; Change
 LOOP UNTIL Change > 0#
 PRINT
 DO
   INPUT "Tolerance for Moment Balance Comparisons (0 to 1) : "; MomTol
 LOOP UNTIL Change > 0#
ELSEIF UCASE$(Response$) = "NF" THEN
   DO
      INPUT "Reset all Fluxes ? ( Y/N ): ", a$
      a$ = UCASE$(LEFT$(LTRIM$(a$), 1))
   LOOP UNTIL INSTR("YN", a$) AND LEN(a$) = 1
   IF a$ = "Y" THEN
      ResetAllFluxes$ = "YES"
   ELSE
      ResetAllFluxes$ = "NO"
   END IF
ELSEIF UCASE$(Response$) = "AS" THEN
 IF TotDifMeth$ = " NONE" THEN TotDifMeth$ = " "
 DO
   PRINT "              Add Spatial Quadrature Method(s) :"
   PRINT
   PRINT "    DD - Diamond Difference           MB - DD,LD,LC,SA,LA,and
EC"
   PRINT "    LD - Linear Discontinuous         EL - Entire List "
   PRINT "    SC - Step Characteristic"
   PRINT "    LC - Linear Characteristic"
   PRINT "    LN - Linear Nodal"
   PRINT "    SA - Step   Adaptive"
   PRINT "    LA - Linear Adaptive"
   PRINT "    EC - Exponential Characteristic"
```

```
    PRINT
    INPUT "Choice? ( DD,LD,SC,LC,LN,SA,LA,EC,MB,EL ): ", DifMeth$
    DifMeth$ = UCASE$(LEFT$(LTRIM$(DifMeth$), 2))
 LOOP UNTIL INSTR("ELMBDDLDSCLCLNSALAEC", DifMeth$) AND LEN(DifMeth$) =
2
 SELECT CASE DifMeth$
    CASE "MB"
       TotDifMeth$ = " DD LD LC SA LA EC "
    CASE "EL"
       TotDifMeth$ = " DD LD SC LC LN SA LA EC "
    CASE ELSE
       IF INSTR(TotDifMeth$, DifMeth$) THEN
       ELSE
          TotDifMeth$ = TotDifMeth$ + LTRIM$(DifMeth$ + " ")
       END IF
 END SELECT
ELSEIF UCASE$(Response$) = "CS" THEN
    TotDifMeth$ = " NONE"
ELSEIF UCASE$(Response$) = "N" AND TotDifMeth$ <> " NONE" THEN
 IF INSTR(TotDifMeth$, "EC") THEN
    CALL SetSwitchECtoLC
    LOCATE 12, 1
    FOR i = 1 TO 12
       PRINT TAB(79); " "
    NEXT i
    LOCATE 12, 1
 END IF
 IF Problemfile$ = "" THEN
    LOCATE 12, 1: PRINT "WARNING: No Problem File is Designated"
    SLEEP (3)
    GOTO Variables
 END IF
 IF UCASE$(Outfile$) = "OFF" THEN
    LOCATE 12, 1: PRINT "WARNING: No Output File is Designated"
    PRINT
    DO
       INPUT "Proceed ? ( Y/N ): ", a$
       a$ = UCASE$(LEFT$(LTRIM$(a$), 1))
    LOOP UNTIL INSTR("YN", a$) AND LEN(a$) = 1
    LOCATE 12, 1
    FOR i = 1 TO 10
       PRINT TAB(79); " "
    NEXT i
    LOCATE 12, 1
    IF a$ = "N" THEN GOTO Variables
 END IF
 GOTO Bottom
END IF
GOTO Variables
Bottom:
'  Open files as Necessary
IF NOT again% THEN
 OPEN Problemfile$ FOR INPUT AS #1
 'check for correct file:
 INPUT #1, Ident$
 LOCATE 9, 38: PRINT Ident$: LOCATE 12, 1
 DO
    INPUT "Correct File? ( Y/N/Quit ): ", a$
    a$ = UCASE$(LEFT$(LTRIM$(a$), 1))
 LOOP UNTIL INSTR("YNQ", a$) AND LEN(a$) = 1
 SELECT CASE a$
    CASE "N"
```

```
      CLOSE #1
      Problemfile$ = ""
      Ident$ = ""
      GOTO Variables
   CASE "Q"
      END
   CASE "Y"
      CLS
   CASE ELSE
      BEEP
      PRINT "ERROR: Unsupported choice in OpenProblemFile."
      STOP
 END SELECT
END IF
' Assign First String toggles and Corresponding Diff Method
ilotogl = 1: ihitogl = 3
DifMeth$ = LTRIM$(MID$(TotDifMeth$, ilotogl, ihitogl))
DifMeth$ = UCASE$(LEFT$(LTRIM$(DifMeth$), 2))
END SUB

SUB BeginSets
SHARED Problemfile$, DifMeth$, QuadMeth$, fixup%, Flfix$, Itermax%
SHARED Change, MomTol, nk, TotDifMeth$, Outfile$, Qoutf$, TKfile$
SHARED Qtkf$, Ident$, again%, NewProblem%, ResetAllFluxes$
SHARED Mom0IntFlag$, MaxMom0rerr, Mom1IntFlag$, MaxMom1rerr
SHARED SwitchEC%, SwECtcLCSetPt
'
'    This provides the user with input values
'    for an initial run.
'
Problemfile$ = "a:test1.IPT"
DifMeth$ = "DD"
TotDifMeth$ = " NONE"
QuadMeth$ = "S"
fixup% = False
Flfix$ = "NO"
Itermax% = 150
Change = .00001#
MomTol = 1E-10
nk = 8
Ident$ = " "
again% = False
NewProblem% = True
SwitchEC% = False
SwECtoLCSetPt = .0005
ResetAllFluxes$ = "YES"
Outfile$ = "off"
Qoutf$ = "n"
TKfile$ = "off"
Qtkf$ = "n"
Mom0IntFlag$ = " 0th: CONSERVED": MaxMom0rerr = 0
Mom1IntFlag$ = " 1st: CONSERVED": MaxMom1rerr = 0
END SUB

DEFINT A-H, J, L-M, O-Z
FUNCTION Choose (prompt$) STATIC
 CONST False = 0
 CONST True = NOT False
 DO
    PRINT prompt$;
    INPUT "", ch$
    ch$ = UCASE$(LEFT$(LTRIM$(ch$), 1))
```

```
 LOOP UNTIL INSTR("YN10", ch$) AND LEN(ch$) = 1
 IF ch$ = "Y" OR ch$ = "1" THEN
    Choose = True
 ELSE
    Choose = False
 END IF
END FUNCTION

DEFSNG H
DEFDBL A-G, J, L-M, O-Z
FUNCTION DdxG (x) STATIC
 ' d/dx g(x) for Newton's Method Root Solving beta in EC
 DdxG = 1 / x ^ 2 - (1 / (EXP(x) - 1)) - (1 / (EXP(x) - 1) ^ 2)
END FUNCTION

FUNCTION Gx (x, ro) STATIC
 'g(x) = p1(beta)/p0(beta) - ro = 0 Root Solved by Newton's Method
 Gx = 1 - (1 / x) + (1 / (EXP(x) - 1)) - ro
END FUNCTION

SUB makeP (p(), e) STATIC
SHARED DifMeth$
'Modified From K. Mathews' Original makeP SUB by G. Sjoden
'Exception: LD Pade exponential inserted in LN model by K. Mathews
'
 CONST imax = 3
 DIM pp(-1 TO imax) AS DOUBLE
 DIM ee AS DOUBLE
 IF LBOUND(p) <> -1 OR UBOUND(p) <> imax THEN STOP
 SELECT CASE DifMeth$
  CASE "LD"
  ee = CDBL(e)
  'Forward Iteration: Use Pade qpprox to EXP(-ee)
  ' (Allows LN algorithm to give LD results)
  pp(-1) = (6 - ee * 2) / (6 + ee * (4 + ee))
  p(-1) = pp(-1)
  pp(0) = (1 - pp(-1)) / ee
  p(0) = pp(0)
  FOR i = 1 TO imax
    pp(i) = (1 - i * pp(i - 1)) / ee
    p(i) = pp(i)
  NEXT i
  CASE ELSE      'All other Spatial Quadratures
  SELECT CASE (e)
    CASE 0
       p(-1) = 1
       FOR i = 0 TO imax
          p(i) = 1 / (i + 1)
       NEXT i
    CASE IS > 0
       ee = CDBL(e)
       pp(-1) = EXP(-ee)
       p(-1) = pp(-1)
       pp(0) = (1 - pp(-1)) / ee
       p(0) = pp(0)
       ife = FIX(e)  'Fix truncates e (does NOT round) to integer value
only
        SELECT CASE (ife - 1)
       CASE IS <= imax                    'Partial Forward Iteration
        FOR i = 1 TO (ife - 1)
         pp(i) = (1 - i * pp(i - 1)) / ee
         p(i) = pp(i)
```

```
    PRINT
    INPUT "Choice? ( DD,LD,SC,LC,LN,SA,LA,EC,MB,EL ): ", DifMeth$
    DifMeth$ = UCASE$(LEFT$(LTRIM$(DifMeth$), 2))
 LOOP UNTIL INSTR("ELMBDDLDSCLCLNSALAEC", DifMeth$) AND LEN(DifMeth$) =
2
 SELECT CASE DifMeth$
    CASE "MB"
       TotDifMeth$ = " DD LD LC SA LA EC "
    CASE "EL"
       TotDifMeth$ = " DD LD SC LC LN SA LA EC "
    CASE ELSE
       IF INSTR(TotDifMeth$, DifMeth$) THEN
       ELSE
          TotDifMeth$ = TotDifMeth$ + LTRIM$(DifMeth$ + " ")
       END IF
 END SELECT
ELSEIF UCASE$(Response$) = "CS" THEN
    TotDifMeth$ = " NONE"
ELSEIF UCASE$(Response$) = "N" AND TotDifMeth$ <> " NONE" THEN
 IF INSTR(TotDifMeth$, "EC") THEN
    CALL SetSwitchECtoLC
    LOCATE 12, 1
    FOR i = 1 TO 12
       PRINT TAB(79); " "
    NEXT i
    LOCATE 12, 1
 END IF
 IF Problemfile$ = "" THEN
    LOCATE 12, 1: PRINT "WARNING: No Problem File is Designated"
    SLEEP (3)
    GOTO Variables
 END IF
 IF UCASE$(Outfile$) = "OFF" THEN
    LOCATE 12, 1: PRINT "WARNING: No Output File is Designated"
    PRINT
    DO
       INPUT "Proceed ? ( Y/N ): ", a$
       a$ = UCASE$(LEFT$(LTRIM$(a$), 1))
    LOOP UNTIL INSTR("YN", a$) AND LEN(a$) = 1
    LOCATE 12, 1
    FOR i = 1 TO 10
       PRINT TAB(79); " "
    NEXT i
    LOCATE 12, 1
    IF a$ = "N" THEN GOTO Variables
 END IF
 GOTO Bottom
END IF
GOTO Variables
Bottom:
'  Open files as Necessary
IF NOT again% THEN
 OPEN Problemfile$ FOR INPUT AS #1
 'check for correct file:
 INPUT #1, Ident$
 LOCATE 9, 38: PRINT Ident$: LOCATE 12, 1
 DO
    INPUT "Correct File? ( Y/N/Quit ): ", a$
    a$ = UCASE$(LEFT$(LTRIM$(a$), 1))
 LOOP UNTIL INSTR("YNQ", a$) AND LEN(a$) = 1
 SELECT CASE a$
    CASE "N"
```

```
      CLOSE #1
      Problemfile$ = ""
      Ident$ = ""
      GOTO Variables
   CASE "Q"
      END
   CASE "Y"
      CLS
   CASE ELSE
      BEEP
      PRINT "ERROR: Unsupported choice in OpenProblemFile."
      STOP
 END SELECT
END IF
' Assign First String toggles and Corresponding Diff Method
ilotogl = 1: ihitogl = 3
DifMeth$ = LTRIM$(MID$(TotDifMeth$, ilotogl, ihitogl))
DifMeth$ = UCASE$(LEFT$(LTRIM$(DifMeth$), 2))
END SUB

SUB BeginSets
SHARED Problemfile$, DifMeth$, QuadMeth$, fixup%, Flfix$, Itermax%
SHARED Change, MomTol, nk, TotDifMeth$, Outfile$, Qoutf$, TKfile$
SHARED Qtkf$, Ident$, again%, NewProblem%, ResetAllFluxes$
SHARED Mom0IntFlag$, MaxMom0rerr, Mom1IntFlag$, MaxMom1rerr
SHARED SwitchEC%, SwECtoLCSetPt
'
'    This provides the user with input values
'    for an initial run.
'
Problemfile$ = "a:test1.IPT"
DifMeth$ = "DD"
TotDifMeth$ = " NONE"
QuadMeth$ = "S"
fixup% = False
Flfix$ = "NO"
Itermax% = 150
Change = .00001#
MomTol = 1E-10
nk = 8
Ident$ = " "
again% = False
NewProblem% = True
SwitchEC% = False
SwECtoLCSetPt = .0005
ResetAllFluxes$ = "YES"
Outfile$ = "off"
Qoutf$ = "n"
TKfile$ = "off"
Qtkf$ = "n"
Mom0IntFlag$ = " 0th: CONSERVED": MaxMom0rerr = 0
Mom1IntFlag$ = " 1st: CONSERVED": MaxMom1rerr = 0
END SUB

DEFINT A-H, J, L-M, O-Z
FUNCTION Choose (prompt$) STATIC
 CONST False = 0
 CONST True = NOT False
 DO
    PRINT prompt$;
    INPUT "", ch$
    ch$ = UCASE$(LEFT$(LTRIM$(ch$), 1))
```

```
 LOOP UNTIL INSTR("YN10", ch$) AND LEN(ch$) = 1
 IF ch$ = "Y" OR ch$ = "1" THEN
    Choose = True
 ELSE
    Choose = False
 END IF
END FUNCTION

DEFSNG H
DEFDBL A-G, J, L-M, O-Z
FUNCTION DdxG (x) STATIC
 ' d/dx g(x) for Newton's Method Root Solving beta in EC
 DdxG = 1 / x ^ 2 - (1 / (EXP(x) - 1)) - (1 / (EXP(x) - 1) ^ 2)
END FUNCTION

FUNCTION Gx (x, ro) STATIC
 'g(x) = p1(beta)/p0(beta) - ro = 0 Root Solved by Newton's Method
 Gx = 1 - (1 / x) + (1 / (EXP(x) - 1)) - ro
END FUNCTION

SUB makeP (p(), e) STATIC
SHARED DifMeth$
'Modified From K. Mathews' Original makeP SUB by G. Sjoden
'Exception: LD Pade exponential inserted in LN model by K. Mathews
'
 CONST imax = 3
 DIM pp(-1 TO imax) AS DOUBLE
 DIM ee AS DOUBLE
 IF LBOUND(p) <> -1 OR UBOUND(p) <> imax THEN STOP
 SELECT CASE DifMeth$
  CASE "LD"
  ee = CDBL(e)
  'Forward Iteration: Use Pade qpprox to EXP(-ee)
  ' (Allows LN algorithm to give LD results)
  pp(-1) = (6 - ee * 2) / (6 + ee * (4 + ee))
  p(-1) = pp(-1)
  pp(0) = (1 - pp(-1)) / ee
  p(0) = pp(0)
  FOR i = 1 TO imax
    pp(i) = (1 - i * pp(i - 1)) / ee
    p(i) = pp(i)
  NEXT i
  CASE ELSE      'All other Spatial Quadratures
  SELECT CASE (e)
    CASE 0
       p(-1) = 1
       FOR i = 0 TO imax
          p(i) = 1 / (i + 1)
       NEXT i
    CASE IS > 0
       ee = CDBL(e)
       pp(-1) = EXP(-ee)
       p(-1) = pp(-1)
       pp(0) = (1 - pp(-1)) / ee
       p(0) = pp(0)
       ife = FIX(e)  'Fix truncates e (does NOT round) to integer value
only
        SELECT CASE (ife - 1)
       CASE IS <= imax                    'Partial Forward Iteration
        FOR i = 1 TO (ife - 1)
         pp(i) = (1 - i * pp(i - 1)) / ee
         p(i) = pp(i)
```

```
        NEXT i
        b# = 1.785714285714286D-02        'Remainder Backward Iteration
        'p26 Starting Value, p27(ee)=1/56 Double Precision
        '    For Single Precision, compute p16 using p17(e)=1/36
        FOR i = 26 TO 5 STEP -1
         b# = (1 - ee * b#) / i
        NEXT i
        FOR i = 4 TO (ife + 1) STEP -1
         b# = (1 - ee * b#) / i
         p(i - 1) = b#
        NEXT i
        CASE IS > imax
        FOR i = 1 TO imax                 'Total Forward Iteration
         pp(i) = (1 - i * pp(i - 1)) / ee
         p(i) = pp(i)
        NEXT i
         END SELECT
    CASE ELSE
       PRINT : PRINT "Negative argument in SUB MakeP"
       PRINT "Execution Halted.": STOP
  END SELECT
 END SELECT
END SUB

FUNCTION MAX (x, y) STATIC
 IF x > y THEN
    MAX = x
 ELSE
    MAX = y
 END IF
END FUNCTION

FUNCTION MIN (x, y) STATIC
 IF x < y THEN
    MIN = x
 ELSE
    MIN = y
 END IF
END FUNCTION

SUB Mom0StepCheck (Fl, Fr, Fa, eps, Sa, DeltaX, mu) STATIC
SHARED MaxMom0rerr, MomTol, Mom0IntFlag$
  'Verify 0th Moment Balance of the Boltzmann transport equation for
each direction mu
  sumML = (Fr - Fl + eps * Fa)
  sumMR = Sa * DeltaX / ABS(mu)
  IF (sumML <> 0) OR (sumMR <> 0) THEN
    Momrerr = ABS(sumML - sumMR) * 2 / (ABS(sumML) + ABS(sumMR))
  ELSE
    Momrerr = 0
  END IF
  IF Momrerr < 2 THEN MaxMom0rerr = MAX(MaxMom0rerr, Momrerr)
END SUB

SUB Mom1StepCheck (Fl, Fr, Fa, Fx, eps, Sx, DeltaX, mu) STATIC
SHARED MaxMom1rerr, MomTol, Mom1IntFlag$
  'Verify 1st Moment Balance of the Boltzmann transport equation for
each direction mu
  sumML = 3 * (Fl - 2 * Fa + Fr) + (eps * Fx)
  sumMR = Sx * DeltaX / ABS(mu)
  IF (sumML <> 0) OR (sumMR <> 0) THEN
    Momrerr = ABS(sumML - sumMR) * 2 / (ABS(sumML) + ABS(sumMR))
```

```
  ELSE
    Momrerr = 0
  END IF
  IF Momrerr < 2 THEN MaxMom1rerr = MAX(MaxMom1rerr, Momrerr)
END SUB

FUNCTION ParseNthWord$ (a$, n) STATIC
    'for a$ containing words separated by spaces
    'returns the n'th word, with word count starting at 1
 i = n
 'strip leading spaces
 b$ = LTRIM$(a$)
 'remove n leading words
 DO WHILE i > 1 AND b$ <> ""
    'strip first word, up to a space character
    k = INSTR(b$, " ")
    IF k THEN
       b$ = RIGHT$(b$, LEN(b$) - k + 1)
    END IF
    'strip leading spaces
    b$ = LTRIM$(b$)
    i = i - 1
 LOOP
 'remove trailing spaces and words
 k = INSTR(b$, " ")
 IF k > 0 THEN b$ = LEFT$(b$, k - 1)
 ParseNthWord$ = b$
END FUNCTION

SUB PrtTK (labl$, vrbl(), istart, n) STATIC
'
'    -> NOTE DEVICE #3 currently used for tk file output <-
'    Because any variable beginning with H will be single precision,
'    any data stored using this routine as is will be stored in single
'    precision.  To get full double precision, change hvalue below back
'    just vrbl(i).
'
 PRINT #3, labl$; ","
 J = 0
 FOR i = istart TO n
   hvalue = vrbl(i)
   J = J + 1
'    if using this sub to read tk generated data, use j<5
'    because tk writes 5 datapoints per line, comma delimited
'    j<4 is used here to accommodate double precision numbers
'    (if used) that could be scrolled across a carriage return
   IF (J < 4) THEN
     PRINT #3, hvalue; ", ";
    ELSE
     PRINT #3, hvalue; ", "
     J = 0
   END IF
 NEXT i
 PRINT #3,
END SUB

SUB SetSwitchECtoLC STATIC
SHARED SwECtoLCSetPt
 PRINT "Set Exponential Characteristic (EC) Default to Linear
Characteristic (LC):"
 PRINT "                Typical Values"
 PRINT "rhox=Sx/(3 Sa)   beta=-(b dx)   [0.5(1-rhox)]=p1(beta)/p0(beta)"
```

```
 PRINT " -or+ 2E-6        +or- 1.2E-5        0.500001   or 0.499999"
 PRINT " -or+ 2.5E-4      +or- 5.0E-4        0.50004167 or 0.49995833"
 PRINT " -or+ 0.32        +or- 2.1           0.66       or 0.34 "
 PRINT "(beta <= 2)  insures  (ABS(Sx) < Sa) and No LC Fixup is used"
 PRINT
 PRINT "Current Default EC to LC Method when ABS(beta) <= ";
 PRINT USING "##.####^^^^"; SwECtoLCSetPt
 INPUT "    Change This ? (Y/N): ", a$
 a$ = UCASE$(LEFT$(LTRIM$(a$), 1))
 IF a$ = "Y" THEN
    DO
       INPUT "New Magnitude of  beta <= 2  for EC Default to LC <= ",
SwECtoLCSetPt
    LOOP UNTIL SwECtoLCSetPt <= 2!
    PRINT USING "    ##.####^^^^"; SwECtoLCSetPt;
    PRINT " Confirmed.": SLEEP (2)
 END IF
END SUB

SUB StepDD (Jin, Jout, Sa, dx, Sigma, mu, Fa) STATIC
 SHARED fixup%
 eps = Sigma * dx / ABS(mu)
 a = eps / 2
 Jout = (Jin * (1 - a) + SGN(mu) * Sa * dx) / (1 + a)
       'balance equation combined with diamond difference assumption
 IF fixup% AND (Jout / mu < 0) THEN
    Jout = 0                                  'negative flux fixup
    Fa = (Sa + ABS(Jin) / dx) / Sigma    'conservation by balance
equation
 ELSE
    Fa = (Jin + Jout) / (2 * mu)         'auxiliary equation (diamond
diff)
 END IF
 Fl = Jin / mu
 Fr = Jout / mu
 CALL Mom0StepCheck(Fl, Fr, Fa, eps, Sa, dx, mu)
END SUB

SUB StepEC (Sigma, DeltaX, mu, Fl, Sa, Sx, Fr, Fa, Fx) STATIC
SHARED SwECtoLCSetPt, SwitchEC%, pi
 'Exponential Characteristic Spatial Quadrature
 DIM pe(-1 TO 3), pb(-1 TO 3), pbee(-1 TO 3)
 eps = Sigma * DeltaX / ABS(mu)
 GOSUB RootSolveBeta
 'Compute Walters functions
 '  Note p(3) values are not required in this scheme
 CALL makeP(pe(), eps)
 CALL makeP(pb(), ABS(beta))
 CALL makeP(pbee(), ABS((beta - eps)))
 'If beta is <0, use transformations of Walters p() functions to avoid
 '  iterations of negative arguments in MakeP SUB
 SELECT CASE beta
    CASE IS >= 0
    CASE ELSE
       Be = EXP(ABS(beta))
       p0 = pb(0): p1 = pb(1): p2 = pb(2)
       pb(0) = p0 * Be
       pb(1) = (p0 - p1) * Be
       pb(2) = (p0 - 2 * p1 + p2) * Be
 END SELECT
 'Note set pbee(0)=( p0(beta-eps) exp(-eps) ),  etc
 'If (beta - eps) is <0, again use transformations of Walters functions,
```

```
'  and note  pbee(1), pbee(2) are not currently needed!
SELECT CASE (beta - eps)
   CASE IS >= 0
      pbee(0) = pbee(0) * pe(-1)
      'NOT NEEDED:  pbee(1) = pbee(1) * pe(-1)
      '             pbee(2) = pbee(2) * pe(-1)
   CASE ELSE
      Be = EXP(ABS(beta - eps))
      p0 = pbee(0)
      pbee(0) = (p0 * Be) * pe(-1)
      'NOT NEEDED:  p1 = pbee(1): p2 = pbee(2)
      '             pbee(1) = ((p0 - p1) * Be) * pe(-1)
      '             pbee(2) = ((p0 - 2 * p1 + p2) * Be) * pe(-1)
END SELECT
'Contributions from Entering Flux (Fl)
Frl = Fl * pe(-1)
Fal = Fl * pe(0)
Fxl = 3 * Fl * eps * (pe(2) - pe(1))
'Contributions from sources
SELECT CASE Sa
   CASE IS > 0
      GOSUB ECQuadrature
   CASE 0
      Frs = 0
      Fas = 0
      Fxs = 0
END SELECT
Fr = Frl + Frs
Fa = Fal + Fas
Fx = Fxl + Fxs
CALL Mom0StepCheck(Fl, Fr, Fa, eps, Sa, DeltaX, mu)
CALL Mom1StepCheck(Fl, Fr, Fa, Fx, eps, Sx, DeltaX, mu)
EndofEC:
EXIT SUB

RootSolveBeta:
SELECT CASE Sa
   CASE IS <> 0
      rhox = Sx / (3 * Sa)
      ro = .5 * (1 - rhox)
   CASE ELSE
      rhox = 0
      ro = .5
END SELECT
'Determine first guess
SELECT CASE ro
   CASE IS < 0!
      ro = .001
      beta0 = -1000
   CASE IS > 1!
      ro = .999
      beta0 = 1000
   CASE IS = .5
      beta0 = 0#
   CASE IS >= .77
      beta0 = 1 / (1 - ro)
   CASE IS > .23
      beta0 = 12 / pi * TAN((ro - .5) * pi)
   CASE IS <= .23
      beta0 = -1 / ro
END SELECT
'Rootsolve for beta = -(b dx) using Newton's Method if beta0 is above
```

```
 ' switchpoint to LC (where beta~0 and closely approximates LC with
Sx<<Sa)
 SELECT CASE ABS(beta0)
    CASE 0
       beta = beta0
    CASE IS > SwECtoLCSetPt
       betatol = .00001
       GOSUB LoopNewton
    CASE ELSE
       SwitchEC% = True
       CALL StepLC(Sigma, DeltaX, mu, Fl, Sa, Sx, Fr, Fa, Fx)
       GOTO EndofEC
 END SELECT
RETURN

LoopNewton:
 icount = 0
 DO
    icount = icount + 1
    beta = beta0 - (Gx(beta0, ro) / DdxG(beta0))
    'PRINT ro; " "; icount; "  "; beta0; "  "; beta
    correct% = ((ABS(beta - beta0) / (ABS(beta0))) < betatol)
    IF NOT correct% THEN beta0 = beta
 LOOP UNTIL (correct%) OR (icount >= 1001)
 IF (icount >= 1001) THEN
    PRINT : PRINT "Newton Iteration Loop in StepEC Exceeded 1000
iterations"
    PRINT "Execution Halted.": PRINT
    STOP
 END IF
RETURN

ECQuadrature:
 'Compute coefficient a (from S(x) = a exp(b x)) and other multipliers
 a = Sa / pb(0)
 cR = a * DeltaX / ABS(mu)
 Ca = cR / eps
 Cx1 = 3 * Ca * beta
 Cx2 = 3 * Ca / eps
 'Flux contributions from sources
 Frs = cR * pbee(0)
 Fas = Ca * (pb(0) - pbee(0))
 Fxs = (Cx1 * (pb(2) - pb(1))) + (Cx2 * (2 * pb(0) - (eps + 2) *
pbee(0)))
RETURN
END SUB

SUB StepLA (Sigma, DeltaX, mu, Fl, Sa, Sx, Fr, Fa, Fx) STATIC
 'Constrained Linear Characteristic Spatial Quadrature
 DIM plt(-1 TO 3), pe(-1 TO 3), pt(-1 TO 3)
 eps = Sigma * DeltaX / ABS(mu)
 'Contributions from Entering Flux (Fl)
 CALL makeP(pe(), eps)
 Frl = Fl * pe(-1)
 Fal = Fl * pe(0)
 Fxl = 3 * Fl * (pe(0) - 2 * pe(1))
 'Contributions from sources
 SELECT CASE Sa
    CASE IS > 0
       GOSUB ConstrainedLinearSourceQuadrature
    CASE 0
       Frs = 0
```

```
      Fas = 0
      Fxs = 0
 END SELECT
 Fr = Frl + Frs
 Fa = Fal + Fas
 Fx = Fxl + Fxs
 CALL Mom0StepCheck(Fl, Fr, Fa, eps, Sa, DeltaX, mu)
 CALL Mom1StepCheck(Fl, Fr, Fa, Fx, eps, Sx, DeltaX, mu)
EXIT SUB

ConstrainedLinearSourceQuadrature:
 a = Sa * DeltaX / mu
 x = Sx * DeltaX / mu
SELECT CASE Sx
 CASE -Sa TO Sa                                  '"WedgeLin"
    Frs = (a - x) * pe(0) + 2 * x * pe(1)
    Fas = (a - x) * pe(1) + x * pe(2)
    Fxs = 3 * (a - x) * (pe(1) - pe(2)) + x * (3 * pe(2) - 2 * pe(3))
 CASE -3 * Sa TO -Sa                             '"WedgeNeg"
    rho = ABS(Sx) / (3 * Sa)
    'PRINT "WedgeNeg", rho
    tau = (1 - rho) * 1.5
    CALL makeP(pt(), eps * tau)
    CALL makeP(plt(), eps * (1 - tau))
    Frs = 2 * a * plt(-1) * (pt(0) - pt(1))
    Fas = 2 * a * (1 - tau) * plt(0) * (pt(0) - pt(1))
    Fas = Fas + a * tau * (2 * pt(1) - pt(2))
    Sum = (1 - tau) * (pt(0) - pt(1)) * (plt(0) - 2 * (1 - tau) *
plt(1))
    Sum = Sum - tau * (1 - 2 * tau) * pt(1)
    Sum = 6 * Sum + 3 * tau * (1 - 4 * tau) * pt(2) + 2 * tau ^ 2 *
pt(3)
    Fxs = a * Sum
 CASE Sa TO 3 * Sa                               '"WedgePos
    rho = ABS(Sx) / (3 * Sa)
    'PRINT "WedgePos", rho
    tau = (1 - rho) * 1.5
    CALL makeP(pt(), eps * tau)
    Frs = 2 * a * pt(1)
    Fas = a * tau * pt(2)
    Fxs = a * tau * (3 * pt(2) - 2 * tau * pt(3))
END SELECT
RETURN
END SUB

SUB StepLC (Sigma, DeltaX, mu, Fl, Sa, Sx, Fr, Fa, Fx) STATIC
 'Linear Characteristic Spatial Quadrature
 DIM plt(-1 TO 3), pe(-1 TO 3), pt(-1 TO 3)
 eps = Sigma * DeltaX / ABS(mu)
 'Contributions from Entering Flux (Fl)
 CALL makeP(pe(), eps)
 Frl = Fl * pe(-1)
 Fal = Fl * pe(0)
 Fxl = 3 * Fl * (pe(0) - 2 * pe(1))
 'Contributions from sources
 SELECT CASE Sa
    CASE IS > 0
       a = Sa * DeltaX / mu
       x = Sx * DeltaX / mu
       Frs = (a - x) * pe(0) + 2 * x * pe(1)
       Fas = (a - x) * pe(1) + x * pe(2)
       Fxs = 3 * (a - x) * (pe(1) - pe(2)) + x * (3 * pe(2) - 2 * pe(3))
```

```
    CASE 0
       Frs = 0
       Fas = 0
       Fxs = 0
 END SELECT
 Fr = Frl + Frs
 Fa = Fal + Fas
 Fx = Fxl + Fxs
 CALL Mom0StepCheck(Fl, Fr, Fa, eps, Sa, DeltaX, mu)
 CALL Mom1StepCheck(Fl, Fr, Fa, Fx, eps, Sx, DeltaX, mu)
END SUB

SUB StepLN (Sigma, DeltaX, mu, Fl, Sa, Sx, Fr, Fa, Fx) STATIC
 'Linear Nodal Spatial Quadrature
 DIM pe(-1 TO 3)
 eps = Sigma * DeltaX / ABS(mu)
 'Contributions from Entering Flux (Fl)
 CALL makeP(pe(), eps)
 Frl = Fl * pe(-1)
 Fal = Fl * pe(0)
 Fxl = 3 * Fl * (pe(0) - 2 * pe(1))
 'Contributions from sources
 SELECT CASE Sa
    CASE IS > 0
       GOSUB LNLinearSourceQuadrature
    CASE 0
       Frs = 0
       Fas = 0
       Fxs = 0
 END SELECT
 Fr = Frl + Frs
 Fa = Fal + Fas
 Fx = Fxl + Fxs
 CALL Mom0StepCheck(Fl, Fr, Fa, eps, Sa, DeltaX, mu)
 CALL Mom1StepCheck(Fl, Fr, Fa, Fx, eps, Sx, DeltaX, mu)
EXIT SUB

LNLinearSourceQuadrature:
 a = Sa * DeltaX / mu
 x = Sx * DeltaX / mu
 'Always use method "WedgeLin"
 'Note -- fixup by scalar flux rotation may be used in main program
 Frs = (a - x) * pe(0) + 2 * x * pe(1)
 Fas = (a - x) * pe(1) + x * pe(2)
 Fxs = 3 * (a - x) * (pe(1) - pe(2)) + x * (3 * pe(2) - 2 * pe(3))
RETURN
END SUB

SUB StepSA (Sigma, DeltaX, mu, Fl, Sa, Sx, Fr, Fa, Fx) STATIC
 'Constrained Step Characteristic Spatial Quadrature
 DIM plr(-1 TO 3), pe(-1 TO 3), pr(-1 TO 3)
 eps = Sigma * DeltaX / ABS(mu)
 'Contributions from Entering Flux (Fl)
 CALL makeP(pe(), eps)
 Frl = Fl * pe(-1)
 Fal = Fl * pe(0)
 Fxl = 3 * Fl * (pe(0) - 2 * pe(1))
 'Contributions from sources
 SELECT CASE Sa
    CASE IS > 0
       GOSUB ConstrainedStepSourceQuadrature
    CASE 0
```

```
      Frs = 0
      Fas = 0
      Fxs = 0
 END SELECT
 Fr = Frl + Frs
 Fa = Fal + Fas
 Fx = Fxl + Fxs
 CALL Mom0StepCheck(Fl, Fr, Fa, eps, Sa, DeltaX, mu)
 CALL Mom1StepCheck(Fl, Fr, Fa, Fx, eps, Sx, DeltaX, mu)
EXIT SUB

ConstrainedStepSourceQuadrature:
 a = Sa * DeltaX / mu
 rho = ABS(Sx) / (3 * Sa)
SELECT CASE Sx
 CASE 0 TO 3 * Sa                          '"StepPos"
    CALL makeP(plr(), eps * (1 - rho))
    Frs = a * plr(0)
    Fas = a * (1 - rho) * plr(1)
    Fxs = 3 * a * (1 - rho) * (plr(1) - (1 - rho) * plr(2))
 CASE -3 * Sa TO 0                         '"StepNeg"
    CALL makeP(plr(), eps * (1 - rho))
    CALL makeP(pr(), eps * rho)
    Frs = a * pr(-1) * plr(0)
    Fas = a * (rho * pr(0) * plr(0) + (1 - rho) * plr(1))
    Sum = (1 - rho) * ((1 - 2 * rho) * plr(1) - (1 - rho) * plr(2))
    Sum = Sum + rho * (pr(0) - 2 * rho * pr(1)) * plr(0)
    Fxs = 3 * a * Sum
END SELECT
RETURN
END SUB

SUB StepSC (Sigma, DeltaX, mu, Fl, Sa, Sx, Fr, Fa, Fx) STATIC
 'Step Characteristic Spatial Quadrature
 DIM plr(-1 TO 3), pe(-1 TO 3), pr(-1 TO 3)
 eps = Sigma * DeltaX / ABS(mu)
 'Contributions from Entering Flux (Fl)
 CALL makeP(pe(), eps)
 Frl = Fl * pe(-1)
 Fal = Fl * pe(0)
 'Contributions from sources
 SELECT CASE Sa
    CASE IS > 0
       a = Sa * DeltaX / ABS(mu)
       Frs = a * pe(0)
       Fas = a * pe(1)
    CASE 0
       Frs = 0
       Fas = 0
 END SELECT
 Fr = Frl + Frs
 Fa = Fal + Fas
 Fx = 0
 CALL Mom0StepCheck(Fl, Fr, Fa, eps, Sa, DeltaX, mu)
END SUB
```

Bibliography

Alcouffe, R. E., E. W. Larsen, W. F. Miller, and B. R. Wienke. "Computational Efficiency of Numerical Methods for the Multigroup, Discrete Ordinates Neutron Transport Equations: The Slab Geometry Case ," Nuclear Science and Engineering, 71: 111-127 (1979).

Barbucci, P., F. DiPasquantonio. "Exponential Supplementary Equations for Sn Methods: The One-Dimensional Case," Nuclear Science and Engineering, 63:179 (1977).

De Barros, R. C., E. W. Larsen. "A Numerical Method for One-Group Slab -Geometry Discrete Ordinates Problems with No Spatial Truncation Error," Nuclear Science and Engineering, 104: 198-208 (1990).

Hill, T. R. "ONETRAN: A discrete Ordinates Finite Element Code for the Solution of the One-Dimensional Multigroup Transport Equation," LA-5990-MS, Los Alamos Scientific Laboratory (1975).

Larsen, E. W. "On Numerical Solutions of Transport Problems in the Diffusion Limit," Nuclear Science and Engineering, 83: 90-99 (1983).

Lathrop, K. D. "Spatial Differencing of the Transport Equation: Positivity vs. Accuracy," Journal of Computational Physics, 4: 475-498 (1969).

Lewis, E. E., and W. F. Miller. Computational Methods of Neutron Transport. New York: John Wiley and Sons, 1984.

Mathews, K. A. "Adaptive Characteristic Spatial Quadratures for Discrete Ordinates Neutral Particle Transport -- The Slab Geometry Case," Transport Theory and Statistical Physics, 19(6): 419-458 (1990).

Minor, Bryan. Private Communication. Air Force Institute of Technology, Wright Patterson AFB OH, 1-4 December 1991.

Morel, J. E., E. W. Larsen. "A Multiple Balance Approach for Differencing the Sn Equations," Nuclear Science and Engineering, 105: 1-15 (1990).

Walters, W. F. "Augmented Weighted-Diamond Form of the Linear-Nodal Scheme for Cartesian Coordinate Systems," Nuclear Science and Engineering, 92: 192-196 (1986).

Walters, W. F., R. D. O'Dell. "Nodal Methods for Discrete Ordinates Transport Problems in (x,y) Geometry," Proceedings of an International Topical Meeting on the Advances in Mathematical Methods for the Solution of Engineering Problems. I:115. Munich, Federal Republic of Germany: American Nuclear Society, April 1981.

Vita

Captain Glenn Eric Sjoden was born on 22 September, 1962 in Buffalo, New York. His family transferred to Houston, Texas in 1974. Following graduation from high school in 1980, Glenn attended Texas A&M University in College Station, Texas. At Texas A&M, he worked at the university's TRIGA reactor facility conducting experiments for researchers. Glenn joined the U.S. Air Force in the spring of 1984 and was retained at Texas A&M to complete his bachelors degree. He graduated in 1984 with a Bachelor of Science in nuclear engineering. Following Officer Training School at Lackland AFB, Texas, he was assigned to the Directorate of Nuclear Technology, Air Force Technical Applications Center (AFTAC), at Patrick AFB, Florida. In 1986, Glenn was assigned to an Education With Industry post with the Department of Energy at the Idaho National Engineering Laboratory (INEL) in Idaho Falls, Idaho. At INEL, he worked at Westinghouse in the Nuclear Chemical Processing Plant and for Argonne National Laboratory at the EBR-II Reactor Fuel Manufacturing Facility. In 1987, he returned to AFTAC in Florida. There, he worked in the Nuclear Materials Division of the Nuclear Technology Directorate, until he was assigned to the School of Engineering at the Air Force Institute of Technology in 1990. Glenn is a member of Alpha Nu Sigma, Tau Beta Pi, and is a licensed professional engineer registered in the State of Florida. He is married to Patricia Danielson.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	March 1992	Thesis

4. TITLE AND SUBTITLE
Exponential Characteristic Spatial Quadrature for Discrete Ordinates Neutral Particle Transport in Slab Geometry

5. FUNDING NUMBERS

6. AUTHOR(S)
Glenn E. Sjoden, Capt, USAF

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Air Force Institute of Technology, WPAFB OH 45433-6583

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
N/A

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

A new discrete ordinates spatial quadrature scheme is presented for solving neutral particle transport problems. This new scheme, called the exponential characteristic method, is developed here in slab geometry with isotropic scattering. This method uses a characteristic integration of the Boltzmann transport equation with an exponential function as the assumed form of the source distribution, continuous across each spatial cell. The exponential source function is constructed to globally conserve zeroth and first spatial source moments and is non-negative. Characteristic integration ensures non-negative fluxes and flux moments. Numerical testing indicates that convergence of the exponential characteristic scheme is fourth order in the limit of vanishingly thin cells. Highly accurate solutions to optically thick problems can result using this scheme with very coarse meshes. Comparing accuracy and computational cost with existing spatial quadrature schemes (diamond difference, linear discontinuous, linear characteristic, linear adaptive, etc.), the exponential characteristic scheme typically performed best. This scheme is expected to be expandable to two dimensions in a straight forward manner. Due to the high accuracies achievable using coarse meshes, this scheme may allow researchers to obtain solutions to transport problems once thought too large or too difficult to be adequately solved on conventional computer systems.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Boltzmann transport equation, discrete ordinates, neutron transport, characteristic spatial quadrature, exponential characteristic method	125
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

GENERAL INSTRUCTIONS FOR COMPLETING SF 298

The Report Documentation Page (RDP) is used in announcing and cataloging reports. It is important that this information be consistent with the rest of the report, particularly the cover and title page. Instructions for filling in each block of the form follow. It is important to ***stay within the lines*** to meet ***optical scanning requirements***.

Block 1. Agency Use Only *(Leave blank)*.

Block 2. Report Date. Full publication date including day, month, and year, if available (e.g. 1 Jan 88). Must cite at least the year.

Block 3. Type of Report and Dates Covered. State whether report is interim, final, etc. If applicable, enter inclusive report dates (e.g. 10 Jun 87 - 30 Jun 88).

Block 4. Title and Subtitle. A title is taken from the part of the report that provides the most meaningful and complete information. When a report is prepared in more than one volume, repeat the primary title, add volume number, and include subtitle for the specific volume. On classified documents enter the title classification in parentheses.

Block 5. Funding Numbers. To include contract and grant numbers; may include program element number(s), project number(s), task number(s), and work unit number(s). Use the following labels:

C	- Contract	**PR**	- Project
G	- Grant	**TA**	- Task
PE	- Program Element	**WU**	- Work Unit Accession No.

Block 6. Author(s) Name(s) of person(s) responsible for writing the report, performing the research, or credited with the content of the report. If editor or compiler, this should follow the name(s).

Block 7. Performing Organization Name(s) and Address(es). Self-explanatory.

Block 8. Performing Organization Report Number. Enter the unique alphanumeric report number(s) assigned by the organization performing the report.

Block 9. Sponsoring/Monitoring Agency Name(s) and Address(es). Self-explanatory.

Block 10. Sponsoring/Monitoring Agency Report Number *(If known)*

Block 11. Supplementary Notes. Enter information not included elsewhere such as: Prepared in cooperation with...; Trans. of...; To be published in.... When a report is revised, include a statement whether the new report supersedes or supplements the older report.

Block 12a. Distribution/Availability Statement. Denotes public availability or limitations. Cite any availability to the public. Enter additional limitations or special markings in all capitals (e.g. NOFORN, REL, ITAR).

- **DOD** - See DoDD 5230.24, "Distribution Statements on Technical Documents."
- **DOE** - See authorities.
- **NASA** - See Handbook NHB 2200.2.
- **NTIS** - Leave blank.

Block 12b. Distribution Code.

- **DOD** - Leave blank.
- **DOE** - Enter DOE distribution categories from the Standard Distribution for Unclassified Scientific and Technical Reports.
- **NASA** - Leave blank.
- **NTIS** - Leave blank.

Block 13. Abstract. Include a brief *(Maximum 200 words)* factual summary of the most significant information contained in the report.

Block 14. Subject Terms. Keywords or phrases identifying major subjects in the report.

Block 15. Number of Pages. Enter the total number of pages.

Block 16. Price Code. Enter appropriate price code *(NTIS only)*.

Blocks 17. - 19. Security Classifications. Self-explanatory. Enter U.S. Security Classification in accordance with U.S. Security Regulations (i.e., UNCLASSIFIED). If form contains classified information, stamp classification on the top and bottom of the page.

Block 20. Limitation of Abstract. This block must be completed to assign a limitation to the abstract. Enter either UL (unlimited) or SAR (same as report). An entry in this block is necessary if the abstract is to be limited. If blank, the abstract is assumed to be unlimited.

★U.S.GPO:1990 O-273-271